LSA Basics for Developers

Training Course

J.Fitzek

01.02.2018

Agenda

- LSA System, LSA @ GSI
- Concepts in LSA
  - Accelerators and Layout, Parameter Hierarchy
  - Contexts
  - Settings
  - Trim
- LSA API and Examples

Agenda

- LSA System, LSA @ GSI
- Concepts in LSA
  - Accelerators and Layout, Parameter Hierarchy
  - Contexts
  - Settings
  - Trim
- LSA API and Examples

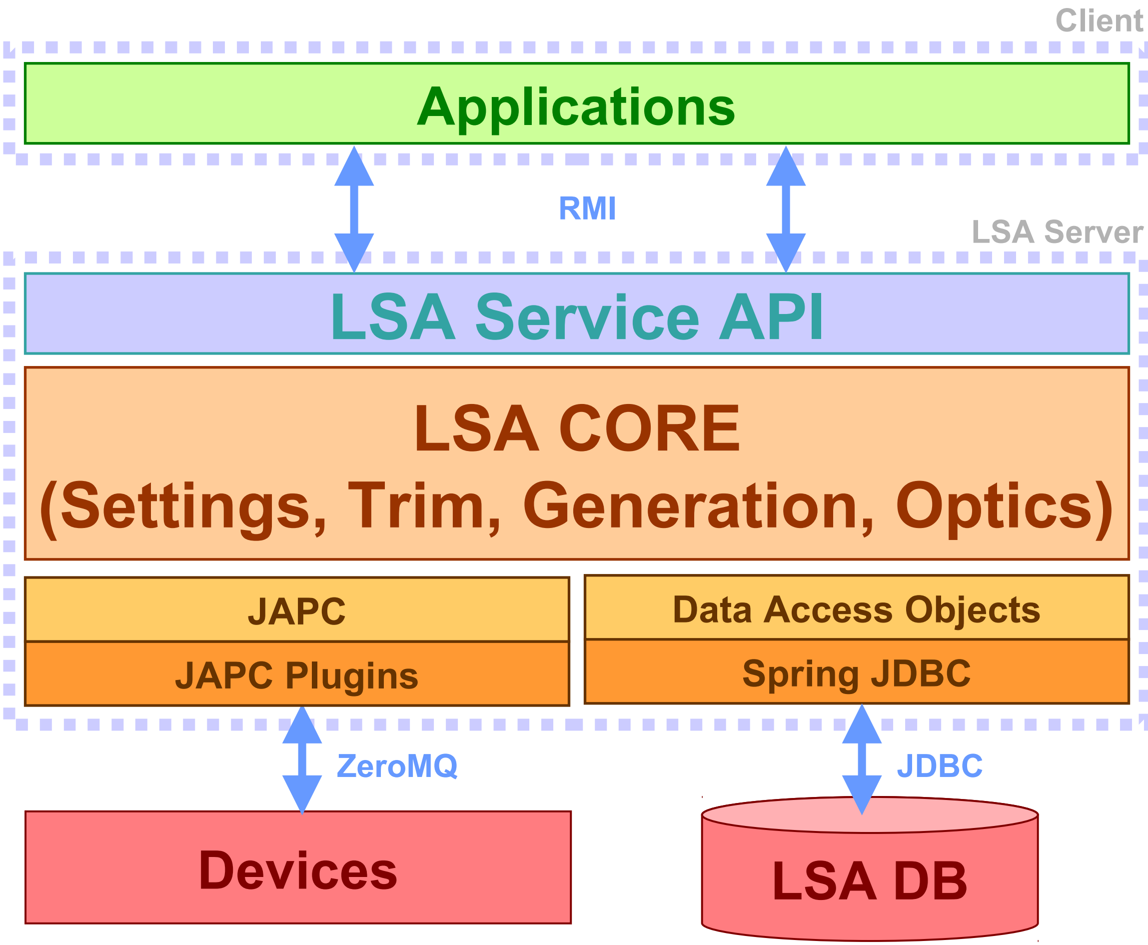
What is LSA? 1/2

- The LSA Project started at CERN in 2001
- LSA was originally called “LHC software architecture”, but is now being used widely at CERN
- Core component of the CERN control system at operating level
- Central part for settings management
- Highly data driven, DB is the master and contains all needed information about optics, devices, contexts, etc
- ONE database schema for all accelerators

What is LSA? 2/2

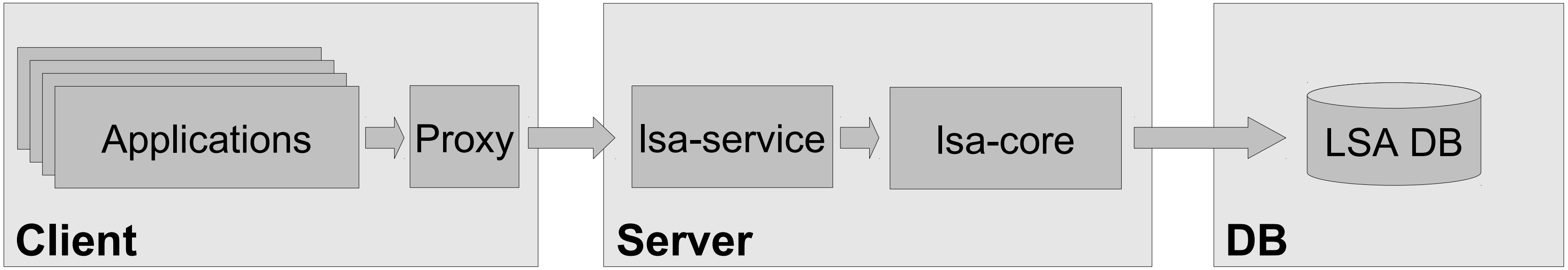
- Parameters are organized in hierarchies
  (from physics to hardware)
- Consistent settings generation and management for all levels
- Devices are accessed using an abstraction layer called JAPC
  that hides middleware specifics
- With LSA, machines are operated on physics level in a consistent way
- Generic approach that allows to just “model” an accelerator within the LSA system and then manage settings for it

LSA Architecture Overview

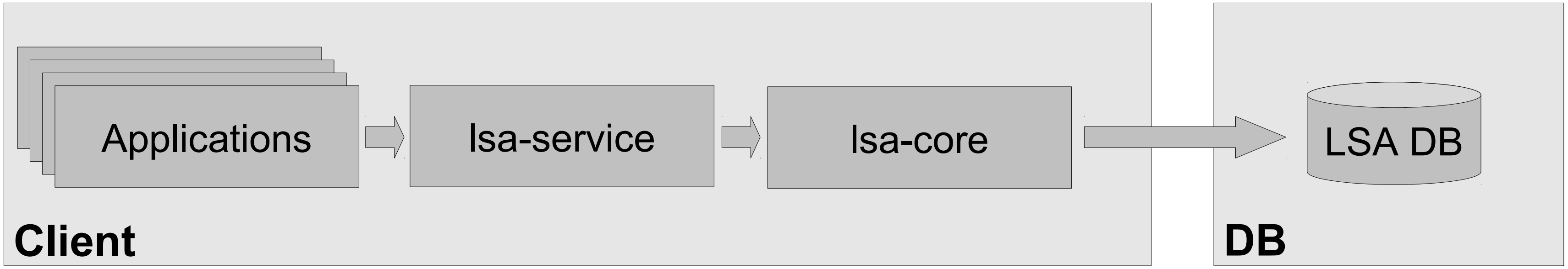


LSA Runtime Setup: 2-tier or 3-tier

- 3-tier Setup: Standard setup for clients, e.g. application development



- 2-tier Setup (for LSA core development)



LSA @ GSI

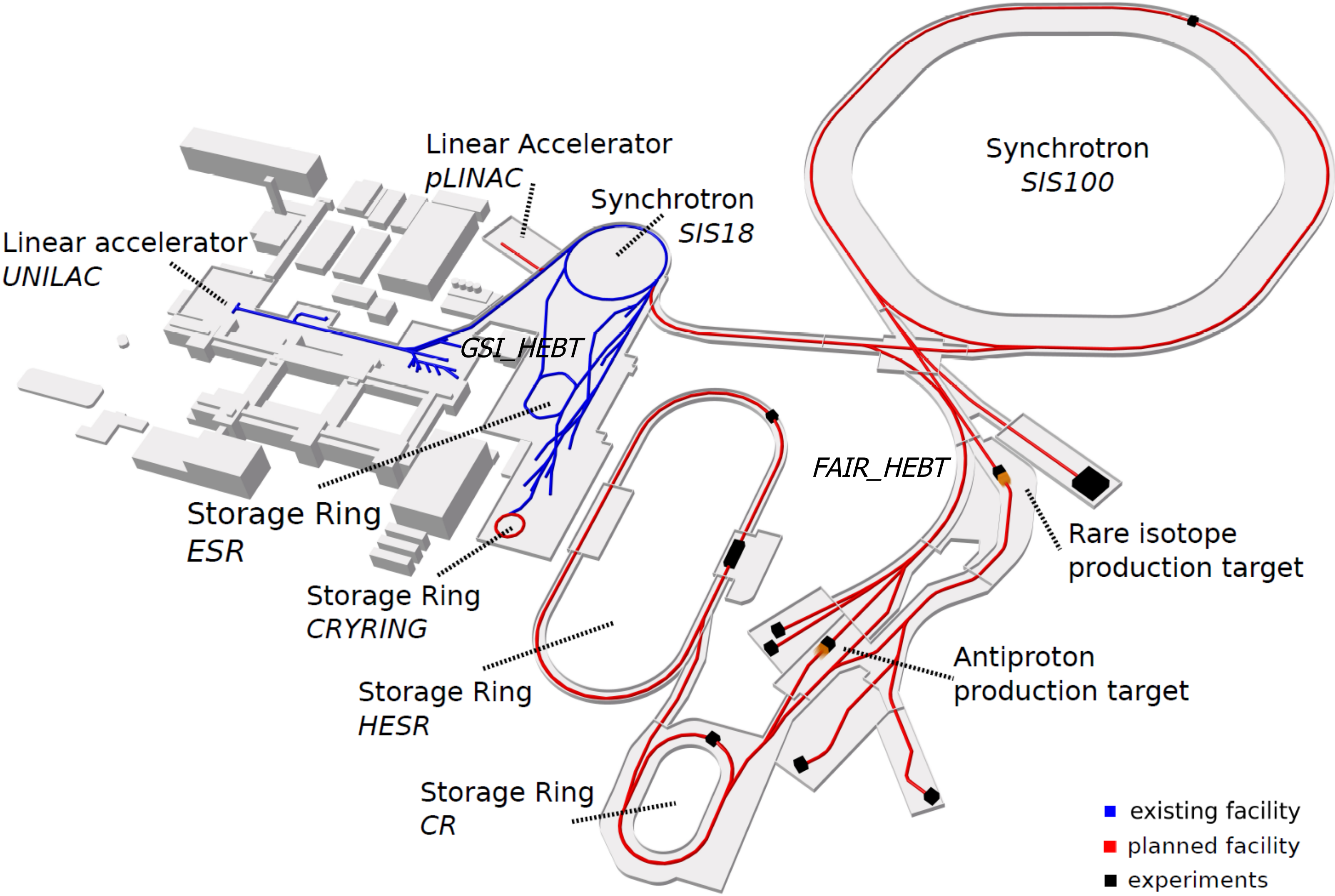
- Project Team at GSI "FAIR Datenversorgung" started end of 2008 (project leader: David Ondreka)
  with members from SYS, ACO, OPE, Machine experts
- Goal is an integrated, homogeneous data supply and settings management for all FAIR accelerators
- As software component for settings management the CERN framework LSA was evaluated and chosen for GSI
- LSA is being enhanced by CERN and GSI in a collaboration since 2008

  => LSA will be the central component for settings management within the new FAIR control system

Agenda

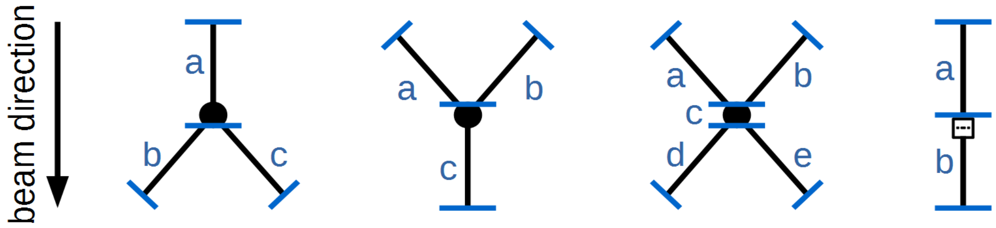
- LSA System, LSA @ GSI
- Concepts in LSA
  - Accelerators and Layout, Parameter Hierarchy
  - Contexts
  - Settings
  - Trim
- LSA API and Examples

LSA Concepts: Accelerator



LSA Concepts: Particle Transfer, AcceleratorZone

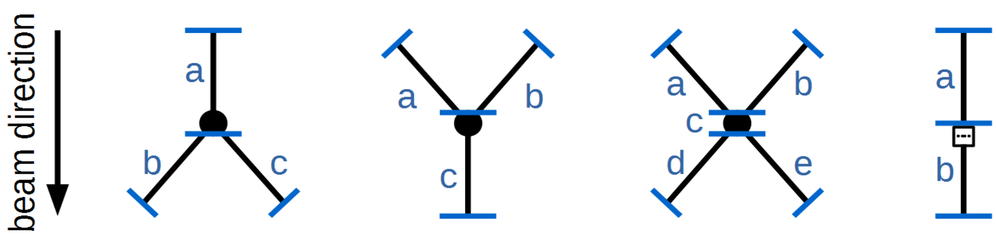
- Particle Transfers are parts of the accelerator with same optics and same timing, typically defined between junction points



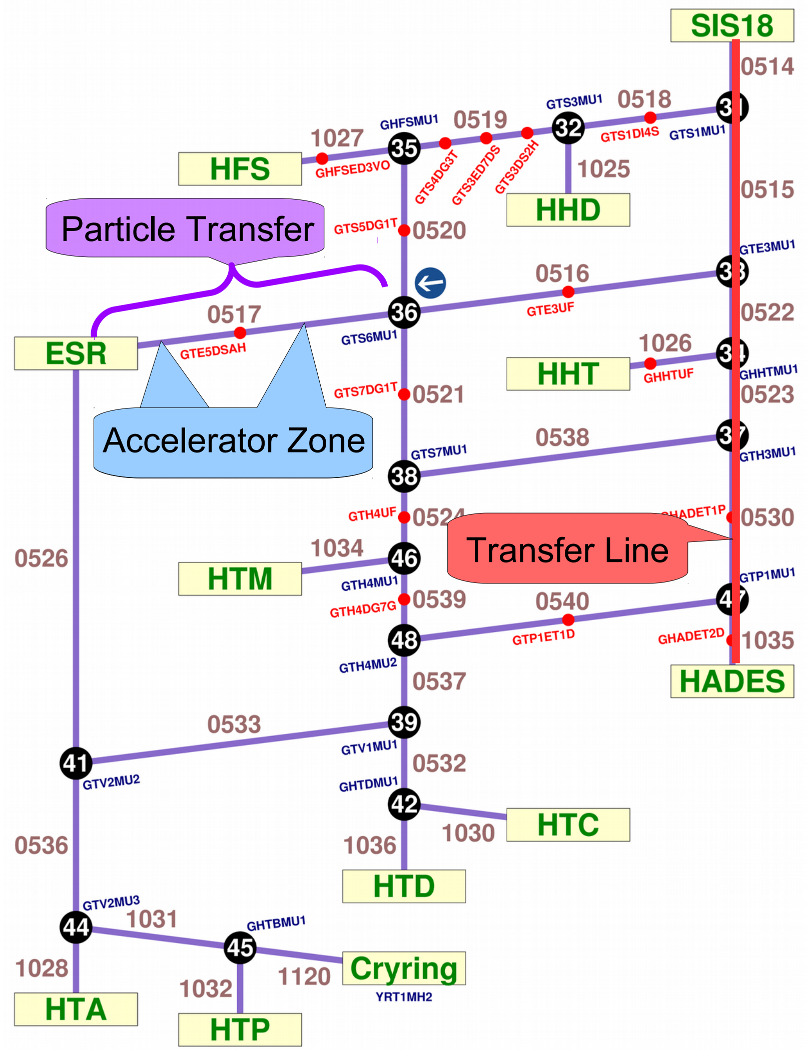
- Particle Transfers can be divided into several Accelerator Zones, if beam attributes (charge..) change, otherwise it is 1:1
- Particle Transfers are grouped into Transfer Lines, e.g. for optics calculation

LSA Concepts: Particle Transfer, AcceleratorZone

- Particle Transfers are parts of the accelerator with same optics and same timing, typically defined between junction points

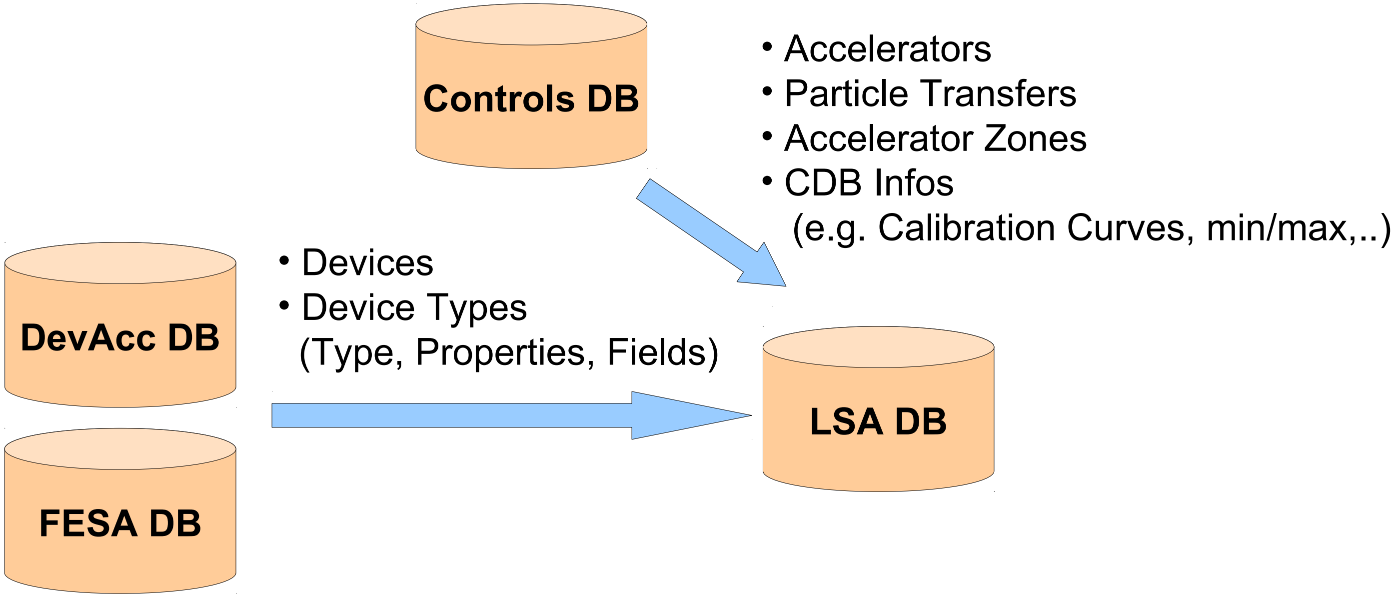


- Particle Transfers can be divided into several Accelerator Zones, if beam attributes (charge..) change, otherwise it is 1:1
- Particle Transfers are grouped into Transfer Lines, e.g. for optics calculation



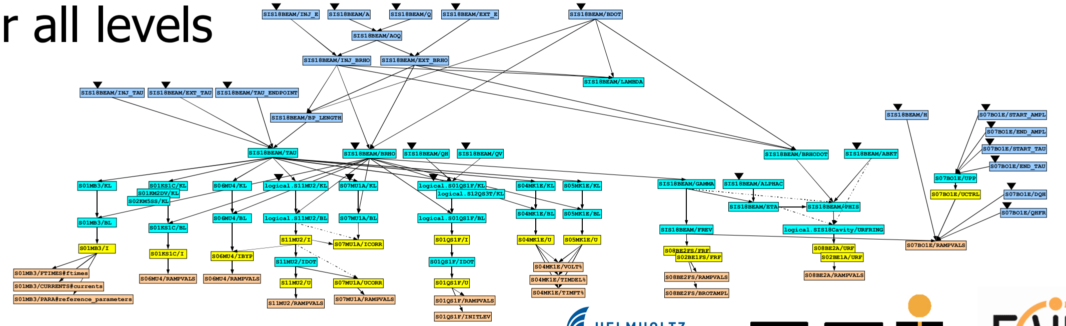
LSA Concepts: Device

- Devices are assigned to Accelerator Zones
- Devices in LSA are imported either from FESA or from DeviceAccess
- Devices have a DeviceType, e.g. BasicPS (for "Basic Power Supply") together with the Properties defined in FESA



LSA Concepts: Parameter Hierarchy

- A parameter defines a settable or measurable entity of the system
- It represents a property of a given device:
  the device can be physical such as a specific power converter,
  logical such as the SIS100 Beam or grouping of equipment
- Parameters are described in a hierarchy
  - represents the logical structure from high-level physics parameters, like beam energy, to machine parameters, like current for a power converter
  - "Roots" are usually changed by operators and are physics parameters
  - "Leaves" are usually hardware parameters
  - Only the lowest parameter level is send to the hardware
  - Settings are kept for all levels

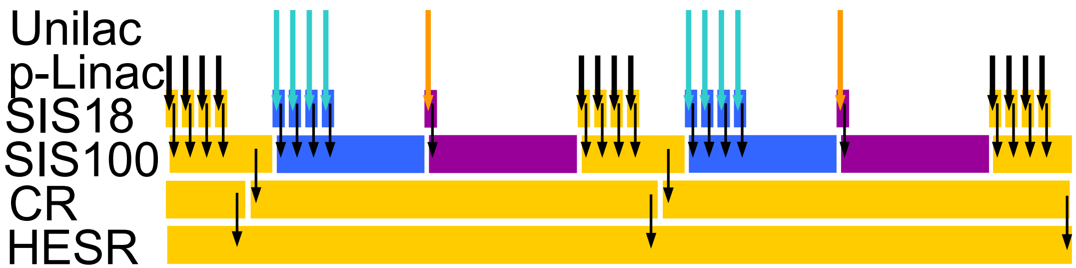


Agenda

- LSA System, LSA @ GSI
- Concepts in LSA
  - Accelerators and Layout, Parameter Hierarchy
  - Contexts
  - Settings
  - Trim
- LSA API and Examples

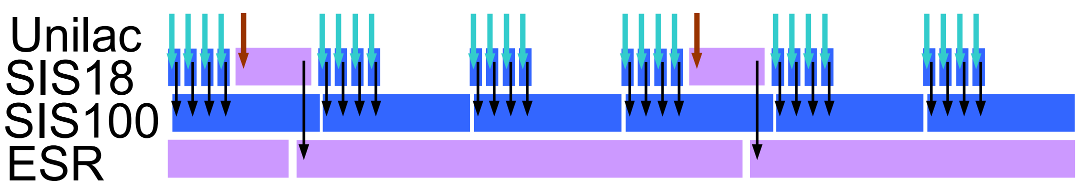
FAIR Operation Scenarios

- Examples for periodic Patterns, each dominated by one main experiment

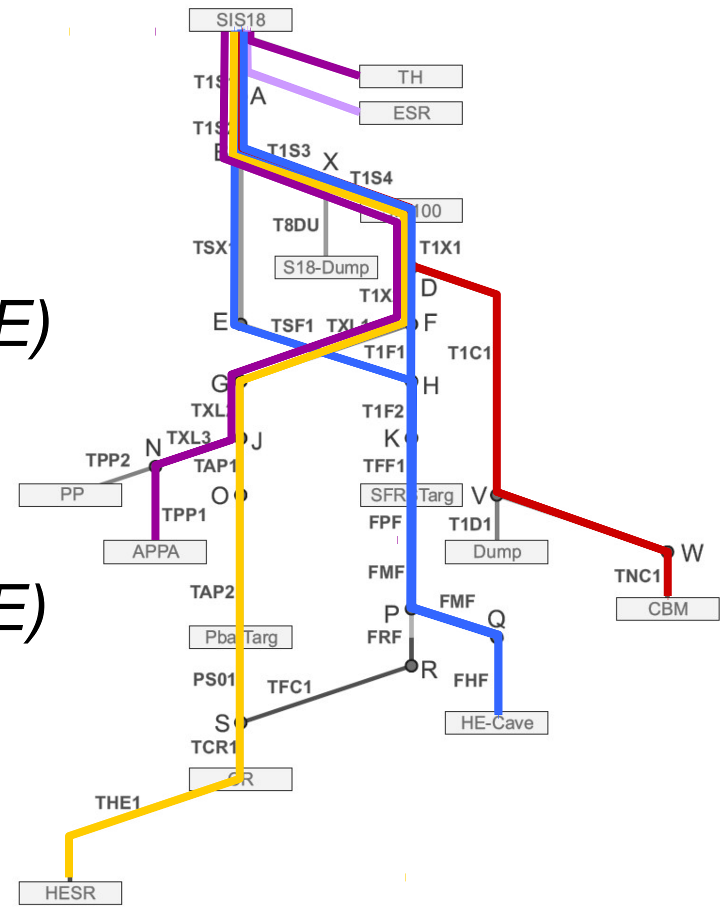


pbar + *RIB ext. target (U^{28+}) + AP (HE)*

CBM + *RIB ext. target (U^{73+}) + AP (LE)*

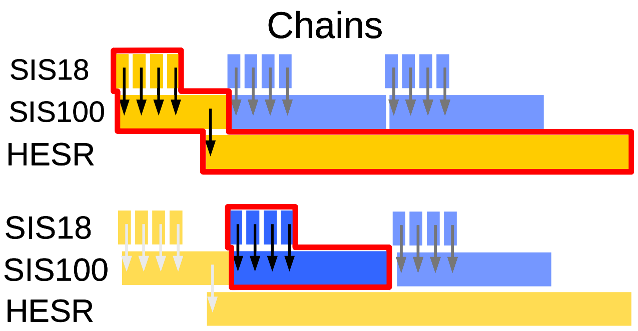


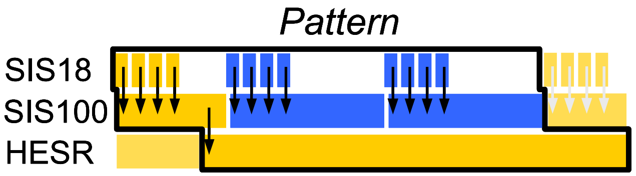
RIB ext. target (U^{28+}) + *ESR*



LSA Concepts: Context (Patterns, Chains)

- Beam Production Chain (BPC)
  - Organisational structure to manage parallel operation and beam transfer through the FAIR accelerator facility
  - Defines sequence and parameters of beam lines from the ion-source up to an experiment
- Pattern
  - Grouping of Beam Production Chains that are executed periodically

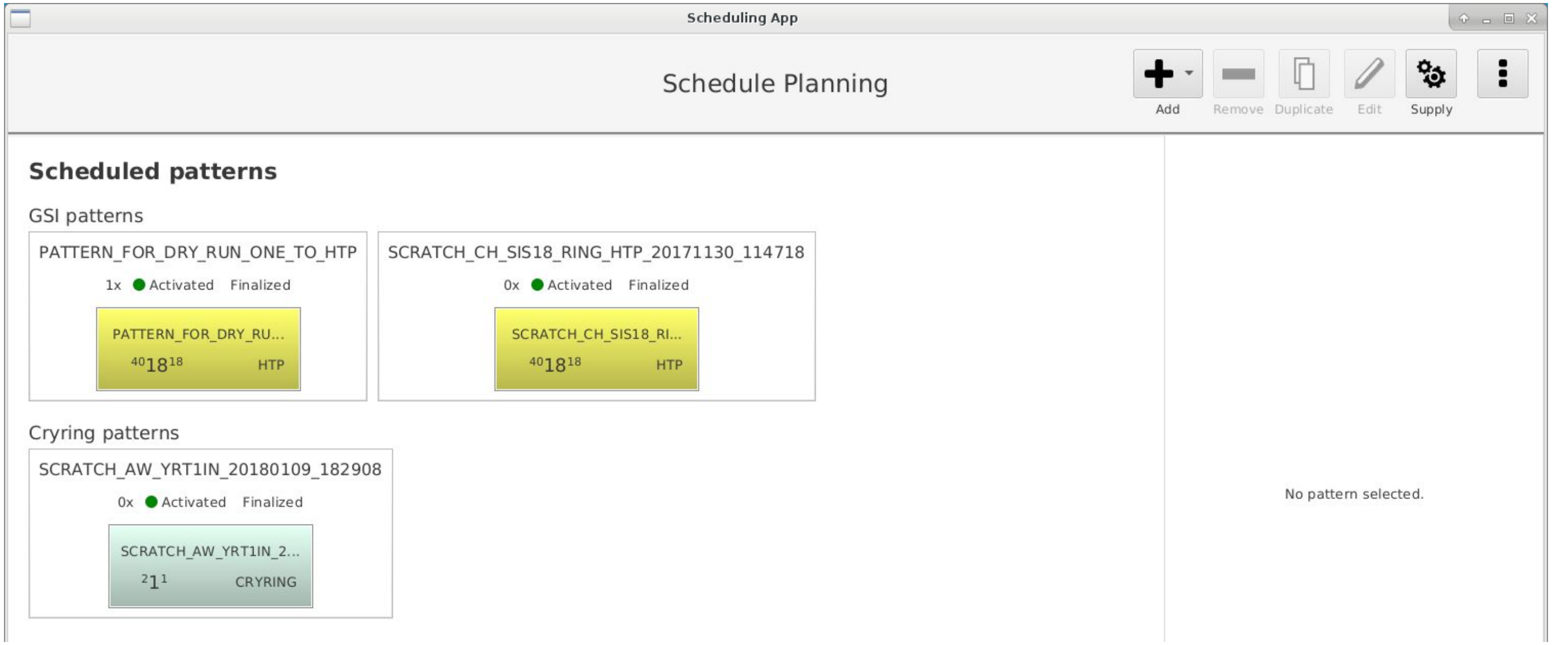




=> For 2018
- One BPC per Pattern
- Multiple Patterns in Round Robin
- Similar to working with Virtual Accelerators

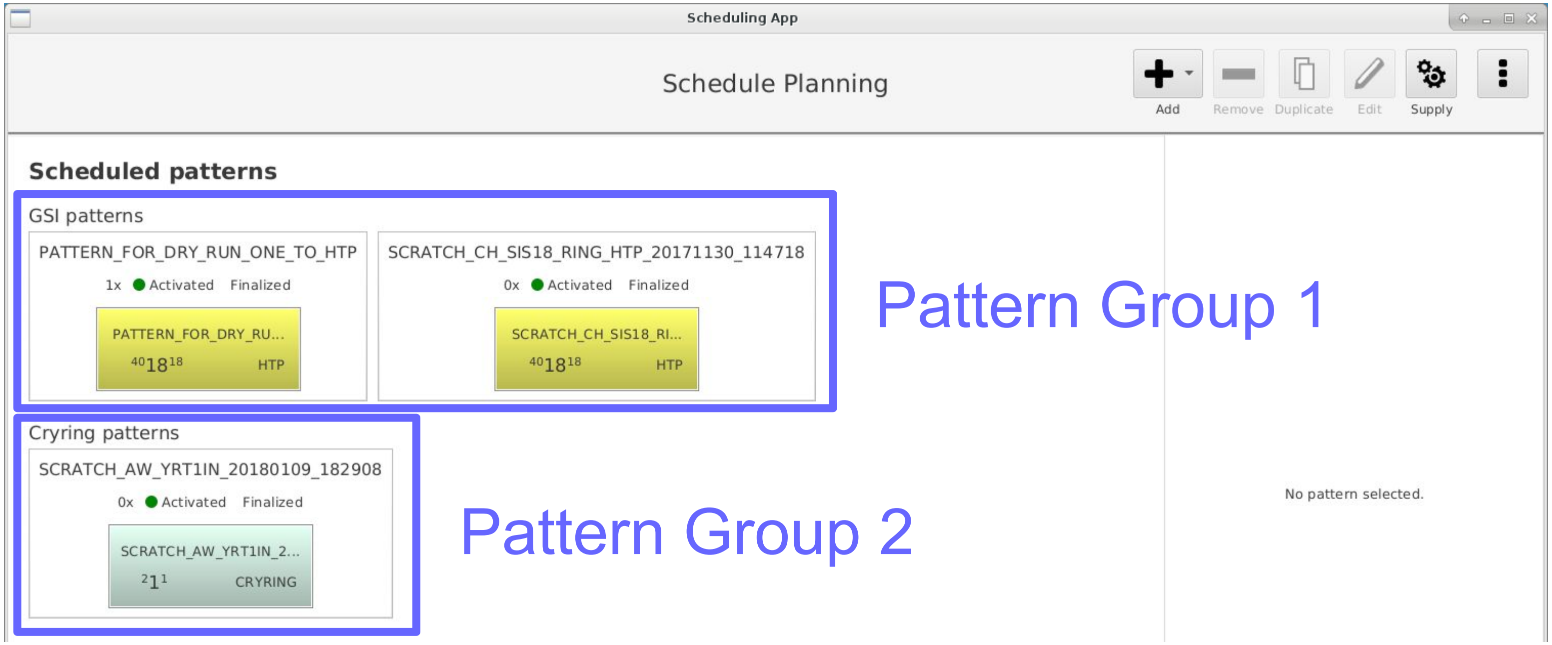
LSA Contexts: Pattern Group

- Several Patterns can be resident in the machine
- Disjoint Patterns can be active at the same time! (e.g. Cryring and SIS18)
- Patterns are therefore organized in Pattern Groups, that contain patterns for different areas of the accelerator complex



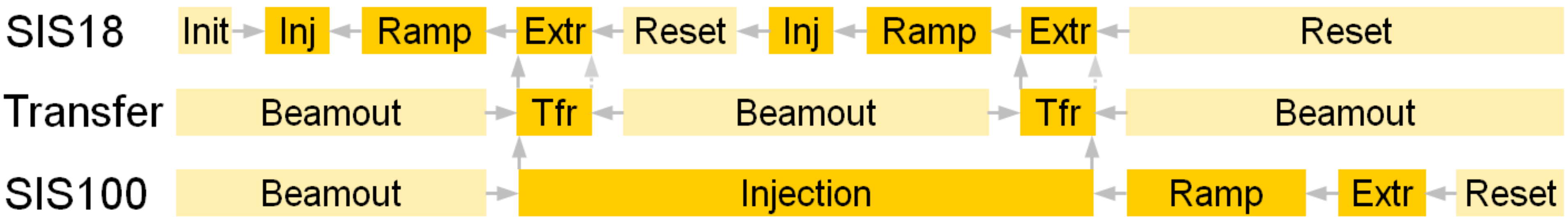
LSA Contexts: Pattern Group

- Several Patterns can be resident in the machine
- Disjoint Patterns can be active at the same time! (e.g. Cryring and SIS18)
- Patterns are therefore organized in Pattern Groups, that contain patterns for different areas of the accelerator complex



LSA Concepts: Context (Beam Processes 1/2)

- Beam Production Chains are composed of Beam Processes
- Beam Processes describe physics processes that are going on in the machine (e.g. injection, ramp, extraction)
- Beam Processes are defined per Particle Transfer
- Settings are kept per Beam Process
- Beam Processes are categorized in BEAM_IN and BEAM_OUT

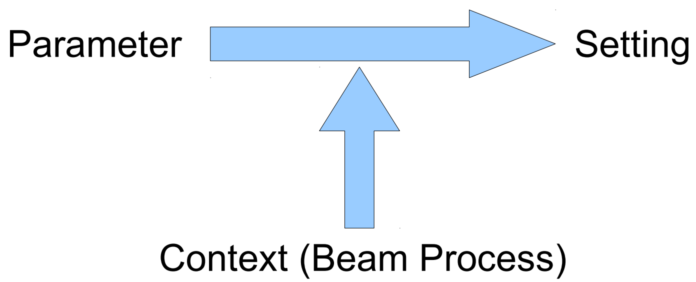


Agenda

- LSA System, LSA @ GSI
- Concepts in LSA
  - Accelerators and Layout, Parameter Hierarchy
  - Contexts
  - Settings
  - Trim
- LSA API and Examples

LSA Concept: Setting 1/2

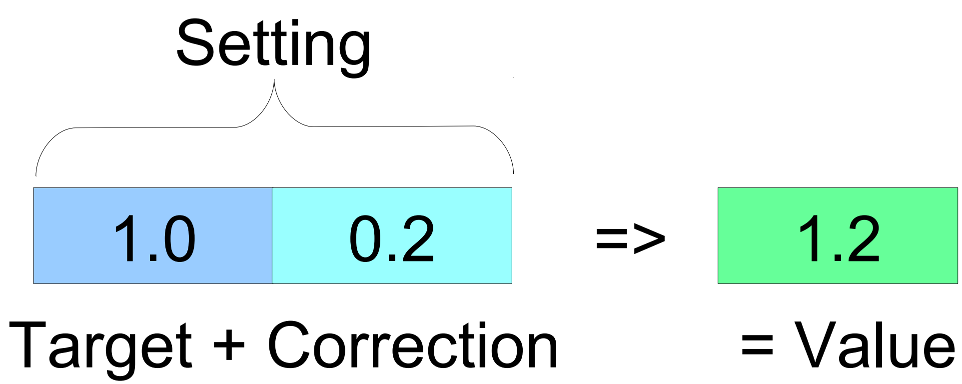
- context in LSA represents something, that can be executed in the machine: BeamProcess, BeamProductionChain, Pattern
- setting is a scalar/function for a parameter depending on a context



- settings are stored for ALL levels of the parameter hierarchy

LSA Concept: Setting 2/2

- a setting consists of target and correction value



- target values are calculated using the accelerator model
- also top level operator input is configured to be placed in target
- => most of the time, *target* is used only, and if necessary, a separate correction-parameter exists
- => correction values are applied only, if the model seems incorrect (expert use, e.g. during commissioning, machine development)

Agenda

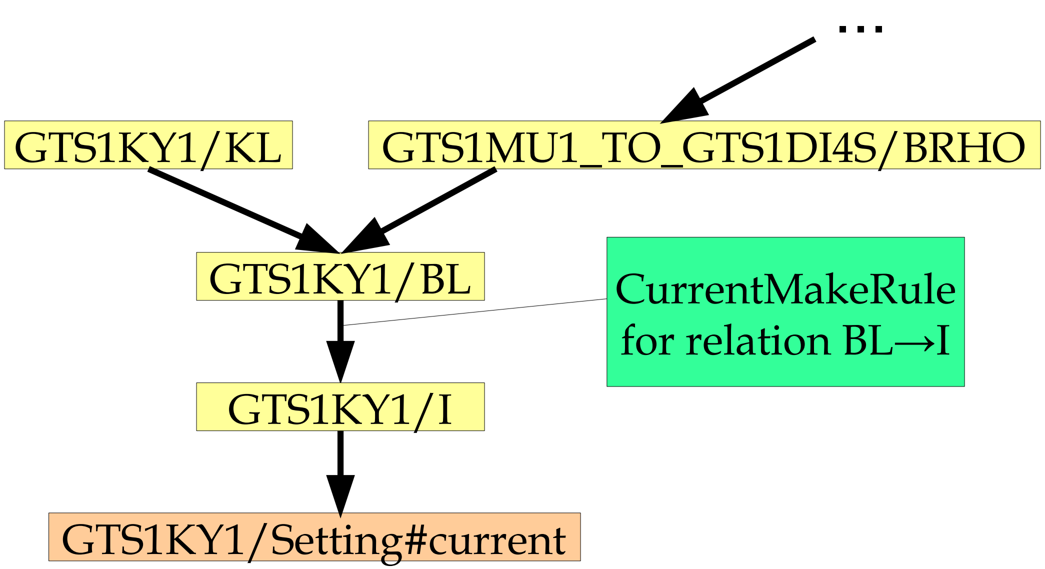
- LSA System, LSA @ GSI
- Concepts in LSA
  - Accelerators and Layout, Parameter Hierarchy
  - Contexts
  - Settings
  - Trim
- LSA API and Examples

What is a trim?

- A trim is a coherent modification of settings,
  plus propagation of changes down the parameter hierarchy
- All applied trims are archived and can be reverted
- The changed settings are also sent to the hardware,
  if the context is *resident*
- The Trim calculation is based on three main concepts:
  - Makerule
  - Linkrule
  - Incorporation Rule

LSA Trim: Makerule 1/2

- The Makerule allows a parameter setting to be computed from its parents
- A Makerule is associated to the relation between two parameter types, typically they are written per *dependent parameter type*
- A trim changing a high level parameter can be automatically propagated down



LSA Trim: Makerule 2/2

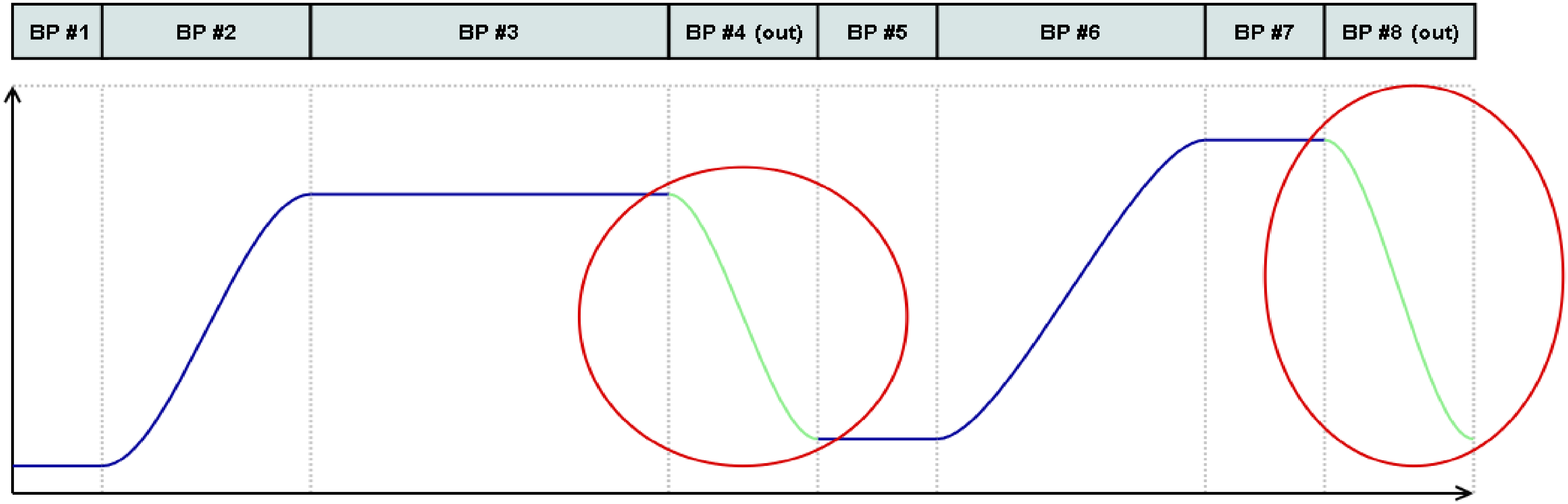
- Example: CurrentMakeRule

▸ lsa-core-gsi ▸ src/main/java ▸ de.gsi.cs.co.lsa.trim.spi.rules.makerule.main ▸ CurrentMakeRule ▸

```java
 */
public class CurrentMakeRule extends AbstractGSIMakeRule {

    @Override
    protected ImmutableValue makeValueImpl(final MakeRuleArguments mra) throws TrimException {

        final BeamProcess beamProcess = mra.getBeamProcess();
        final Parameter srcParameter = getUniqueParentParameter(mra);
        final Parameter depParameter = mra.getParameter();


        final Setting blSetting = getSettingOrNull(mra, srcParameter);

        final ImmutableValue bl = blSetting.getValue();
        final ImmutableValue current;

        final LogicalHardware logicalHardware = findLogicalHardware(depParameter.getDevice().getName());
        final int calSign = logicalHardware.getCalibrationSign();

        current = applyCalibrationFunction(depParameter, beamProcess, CalibrationFunctionTypes.MAG_INTFIELD2CURRENT,
                Operations.multiply(bl, calSign));

        return current;
    }
```

LSA Trim: Linkrules

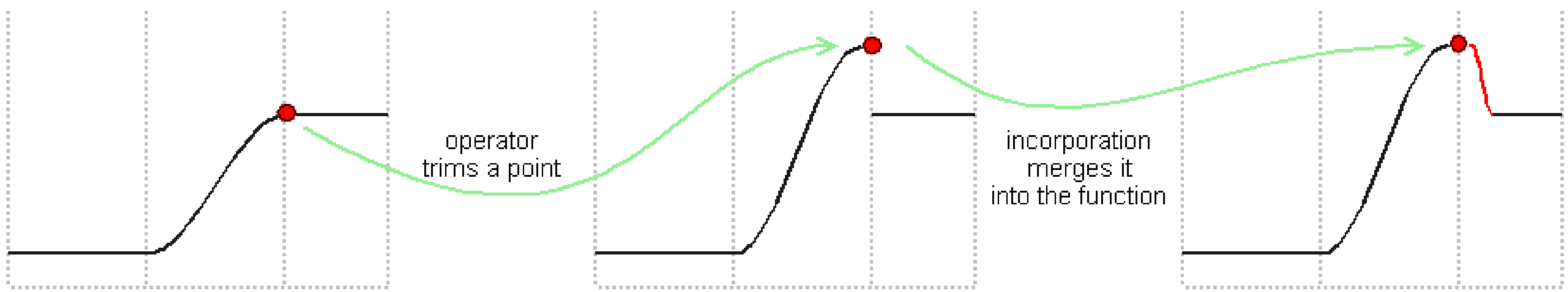
- Link rules generate the link between two beam-in beamprocesses
- Generate settings for the part of the pattern without beam
- Physics parameters only exist when there is beam, hardware parameters are always there, thus Linkrules are only used for hardware parameters (e.g. current)



LSA Model Overview - LSA TEAM - © CERN 2004

LSA Trim: Incorporationrule

- Ensures continuity of functions within the pattern
- Propagates changes from one beam process to its neighbours
- Defined per beamprocess type



Agenda

- LSA System, LSA @ GSI
- Concepts in LSA
  - Accelerators and Layout, Parameter Hierarchy
  - Contexts
  - Settings
  - Trim
- LSA API and Examples

LSA Setup and Usage

- Set up of LSA:
  - definition of the accelerator, import of devices, import optics
  - model the parameter hierarchy and define propagation rules
- Using LSA:
  - Defining e.g. a new pattern calculating all values,
    the values are then stored in the database
  - Drive new settings to the hardware, execute the new cycle
  - Trim values and send the changed settings to the hardware,
    all trims are stored in the database
  - Monitor device values
  - Supply other systems with necessary information about settings
  - **API for applications to get information from the settings management system and to perform trims**

LSA API Examples 1/3

- Convenience Class Services to retrieve LSA Services, e.g.
  - Services.getAcceleratorService()
- AcceleratorService
  - Set<Accelerator> findAccelerators();
  - Set<TransferLine> findTransferLines();
- DeviceService
  - Set<Device> findDevices(DevicesRequest request);
- ParameterService
  - Set<Parameter> findParameters(ParametersRequest request);
  - Set<ParameterTreeNode> findParameterTrees(
    ParameterTreesRequest request);

LSA API Examples 2/3

- ContextService
  - Pattern findPattern(String name);
  - Set<Pattern> findPatterns();
  - Set<Pattern> findResidentPatterns();
  - Set<PatternGroup> findPatternGroups();
- SettingService
  - ContextSettings findContextSettings(StandAloneContext context);
- TrimService
  - TrimResponse trimSettings(TrimRequest request) throws TrimException, DriveException;
  - TrimResponse revertTrim(RevertTrimRequest request) throws ..
  - TrimResponse copySettings(CopySettingsRequest copyRequest) throws ..
- And others:
  - OpticService: find optics, twiss, ion optical elements,..

Contacts

- LSA Team (ACO APP):



Raphael Müller (LSA Projektleiter) | Andreas Schaller | Hanno Hüther

- FAIR Datenversorgung / FAIR data supply:
  - David Ondreka (SYS)
- Papers:
  - ICALEPCS 2017: First Production Use of the new Settings Management System for FAIR
  - PCaPAC 2010: Settings Management within the FAIR control system based on the CERN LSA Framework