Karlsruhe Institute of Technology

Accelerator Technology Platform @ KIT

Institute for Beam Physics and Technology

Management of EPICS IOCs in a distributed network environment using Salt

E. Blomley*, J. Gethmann, M. Schuh, A.-S. Müller, KIT
S. Marsching, aquenos GmbH

Motivation

An EPICS-based control system typically consists of many individual IOCs, which can be distributed across many computers in a network. Managing hundreds of deployed IOCs, keeping track of where they are running, and providing operators with basic interaction capabilities can easily become a maintenance nightmare.

Environment

The two accelerators FLUTE und KARA each operate in separate, self-sufficient network environments. Most EPICS IOCs run on virtual machines using Ubuntu LTS. This requires most hardware being able to communicate via TCP or UDP, with serial communication being managed via serial-to-ethernet hardware gateways. For critical systems, EPICS integrated PLCs are used.

Control System in Numbers

Number of	KARA	FLUTE
EPICS VMs	4	2
EPICS Server	1	2
IP Devices	707	341
PLCs	31	11
Deployed IOCs	94	42
Deployed Services	11	10
Process Variables	~80,000	~12,000

Implementation: Use Salt and one single source of truth file to deploy, maintain and monitor EPICS IOC across distributed hosts.

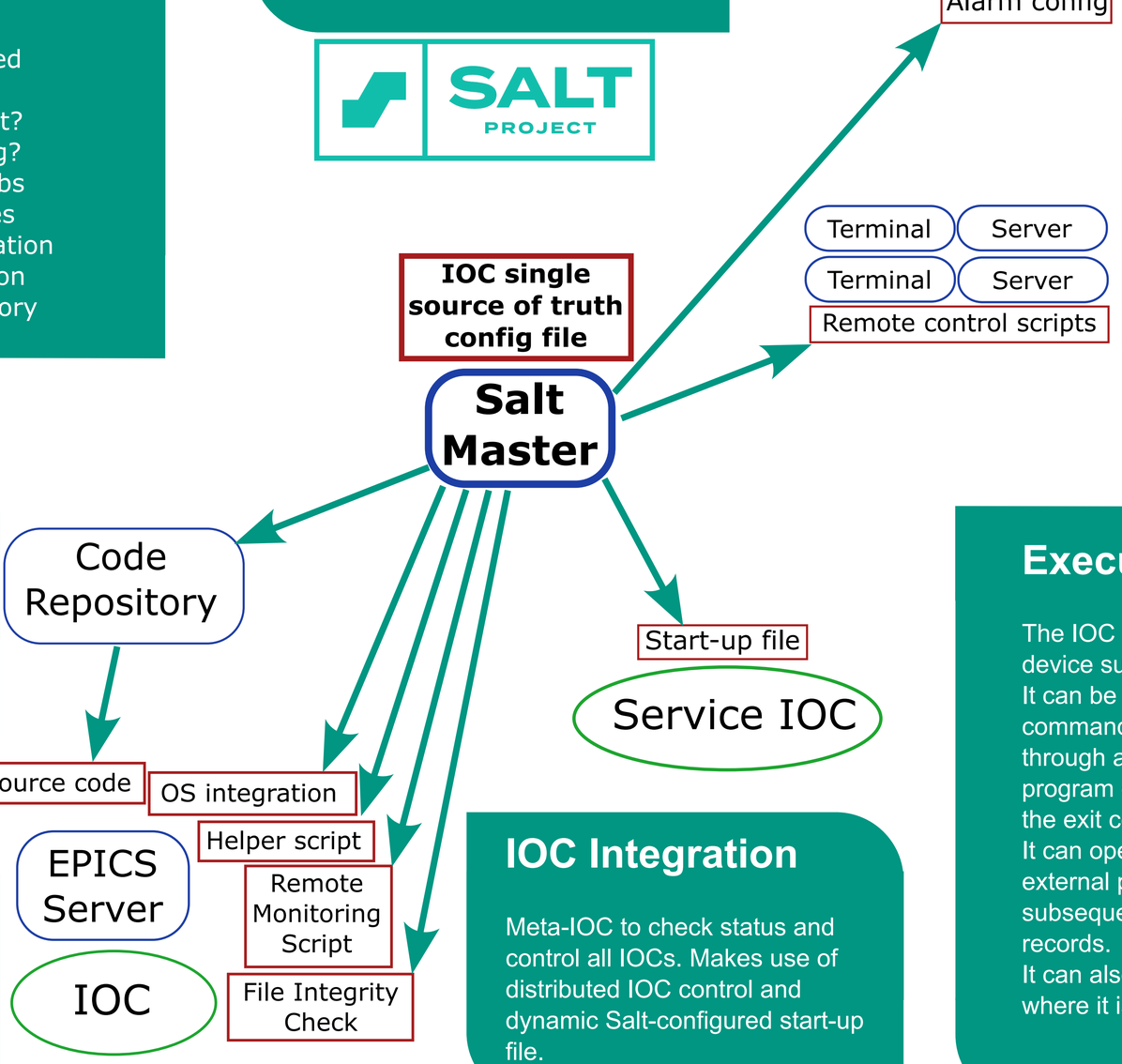
IOC Configuration

Parameter	Description
IOC name	Unique Identifier
Git name	Repository name
Host	Target server
Run script	Default: run
Group	FLUTE, KARA, shared
Type	C or Python
Auto start	Start on host reboot?
Critical	Alarm if not running?
Dependency	For non-default .debs
Git Branch	Testing new features
Docus	External documentation
Description	Short IOC description
Repository	Non-default repository

Salt

Python-based, open-source software for IT automation, remote task execution and configuration management following the infra-structure as code approach.

SALT PROJECT



EPICS Alarming

Automatically configure alarm server to monitor status of IOCs which are configured as critical for accelerator operation.

Distributed IOC Control

Helper script allows control of any IOC from any location using a unique identifier. Script executes command via SSH, knowing on which host the IOC is running.

GitLab Repository

IOCs are organised on a project level grouped by accelerator. A group of shared IOCs exists, implementing accelerator specific start-up files.

Server Deployment

Each IOC is integrated in the host OS, in our case via systemd. In addition, scripts for file integrity, remote monitoring and local control are set up automactially.

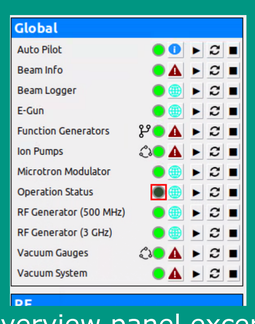
IOC Integration

Meta-IOC to check status and control all IOCs. Makes use of distributed IOC control and dynamic Salt-configured start-up file.

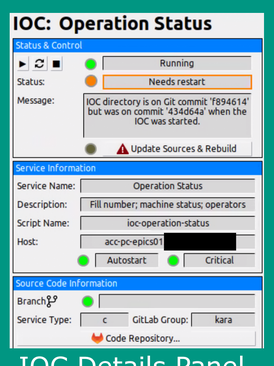
Execute Device Support

The IOC integration makes use of the Execute device support, developed on behalf of KIT.
It can be used to safely run any external commands. Data from EPICS can be passed through and returned values from the executed program can be passed back to EPICS through the exit code or the standard (error) output.
It can operate in a mode where it waits for the external program to finish execution and subsequently triggers the processing of other records.
It can also operate in a fire and forget mode where it is not waiting for the command to finish.

GUI



Overview panel excerpt



IOC Details Panel

Future Plans

Future plans involve automation of the initial IOC creation, continuous integration for IOC code integrity, automated GUI creation, support for device-embedded IOCs and potential steps towards fully containerized deployment, as the general structure would allow for a drop-in replacement of the current EPICS server integration.

References

Salt Project:
https://saltproject.io/

Execture Device Support:
https://github.com/KIT-IBPT/epics-execute

KIT GitLab Repository:
https://gitlab.kit.edu/kit/ibpt/epics

Summary: Consistent and scalable fully automated IOC deployment & integration without any required IOC adjustments, making it also usable for non-IOC services.

E-mail: edmund.blomley@kit.edu