HelixIPLM

Automating the Integration Workflow with IP-Centric Design

Simon Butler, General Manager, Helix IPLM by Perforce

PERFORCE

- Project BoM
- Metadata
- Workspace Management
- Planning
- Release Management
- API for Automation, Integrations

IP / Block level

- Version Control DM at Scale
- Security and Audit History
- Distributed Teams
- Code Review
- Atomic Commits

File level

PERFORCE

SoC Integration Challenges – Distributed Design Teams

Distributed Teams

- Teams are getting more geographically spread out
- SoC complexity dictates multiple design sites
- More external IPs, more reused blocks across multiple designs
- Multiple time zones and difficult coordination across the globe

SoC Integration Challenges – Large Design Data

Design size explosion

- Smaller nodes cause major increase in the size of design data
- 100’s of GB to multiple TB design blocks
- Generating, storing and transferring these blocks strain the network
- Sending data across the globe takes hours, and every change causes multiple delays

SoC Integration Challenges – Lack of Local Expertise

Complex designs necessitate IP & design reuse

- More external IPs and more reuse of existing, verified blocks
- Sourcing of design blocks from multiple geographies / design teams
- Integrators lack local expertise when integrations problems arise
- Leads to long turn-arounds, messy email and spreadsheet-based debugging, missed deadlines

An IP-Centric Approach To Design and Integration

Solve SoC integration challenges with an IP-centric approach to design

- Model each design block as an IP
  - Internal blocks, externally acquired IPs, reused design blocks from other designs – they can all be treated as IPs
- Build a lifecycle for IPs
  - Each block follows the same general lifecycle
  - Aliases and IP Lifecycle stage-gates enable the lifecycle for each IP
  - Sample lifecycle of a design block
- Integrate these IPs into the SoC based on quality enforced by lifecycle rules
  - Allows for automation, which leads to easier integrations with fewer issues

What is IP?

IP is the abstraction of data files that defines an implementation and meta-data that defines its state.

Software, Firmware, & Hardware Design Files

Dependencies, Permissions, Hierarchy, Properties, Usage, etc.

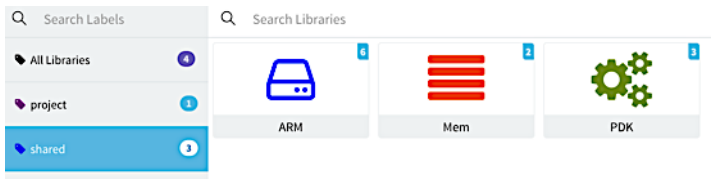
Also referred to as **Block** or **Module**

Annotate IPs with Rich Metadata

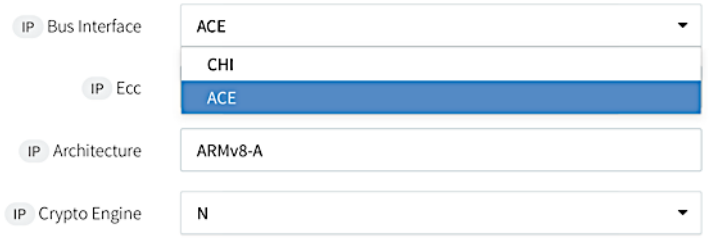
IP Design Data
(In DM)

IPLM IP Object

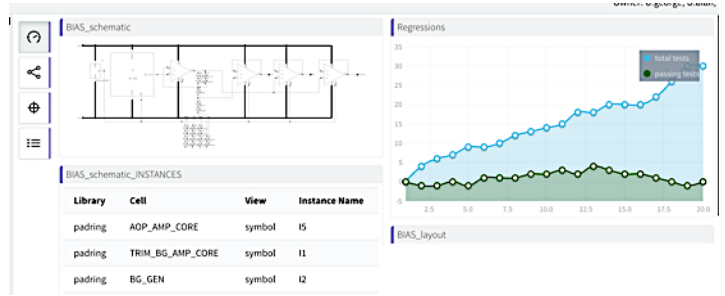
LABELS
For Taxonomy



PROPERTIES
For technical specifications



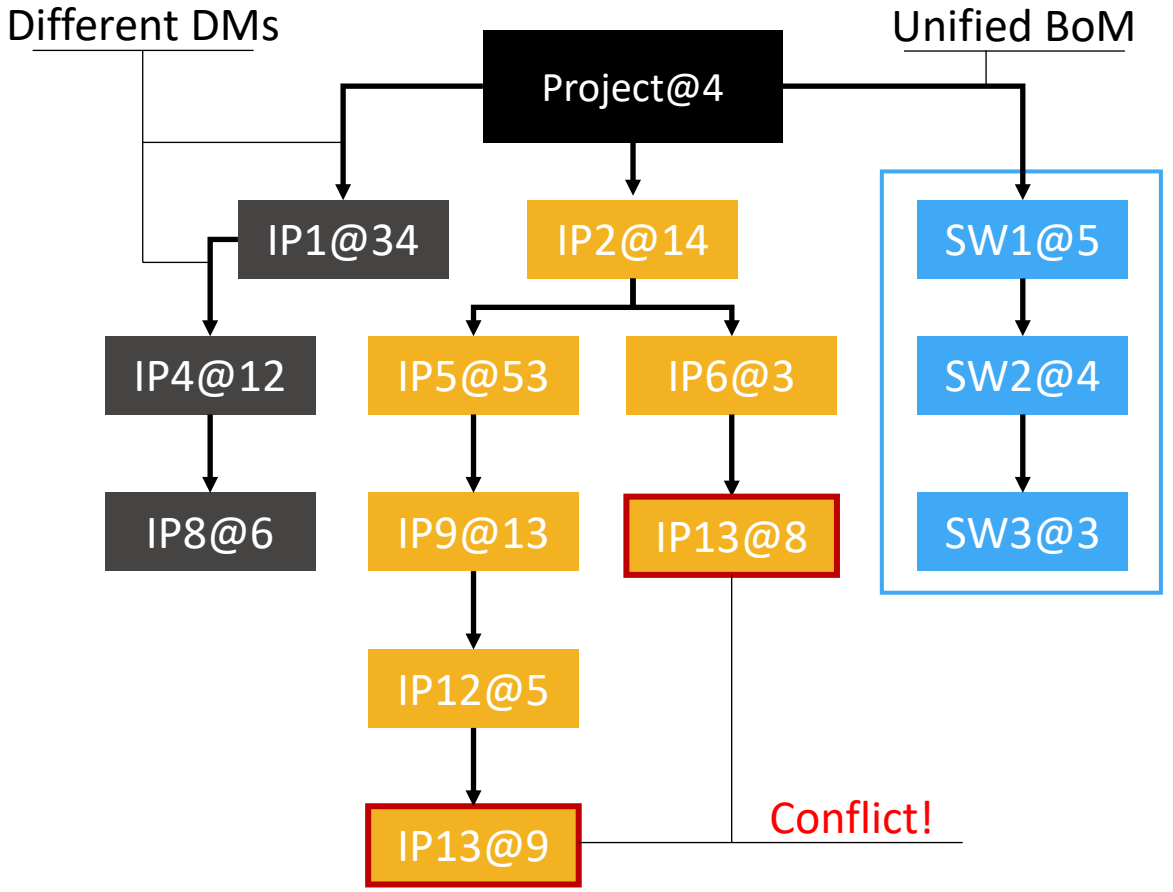
ATTRIBUTES
For Visual Data, Graphs, Tables, etc.



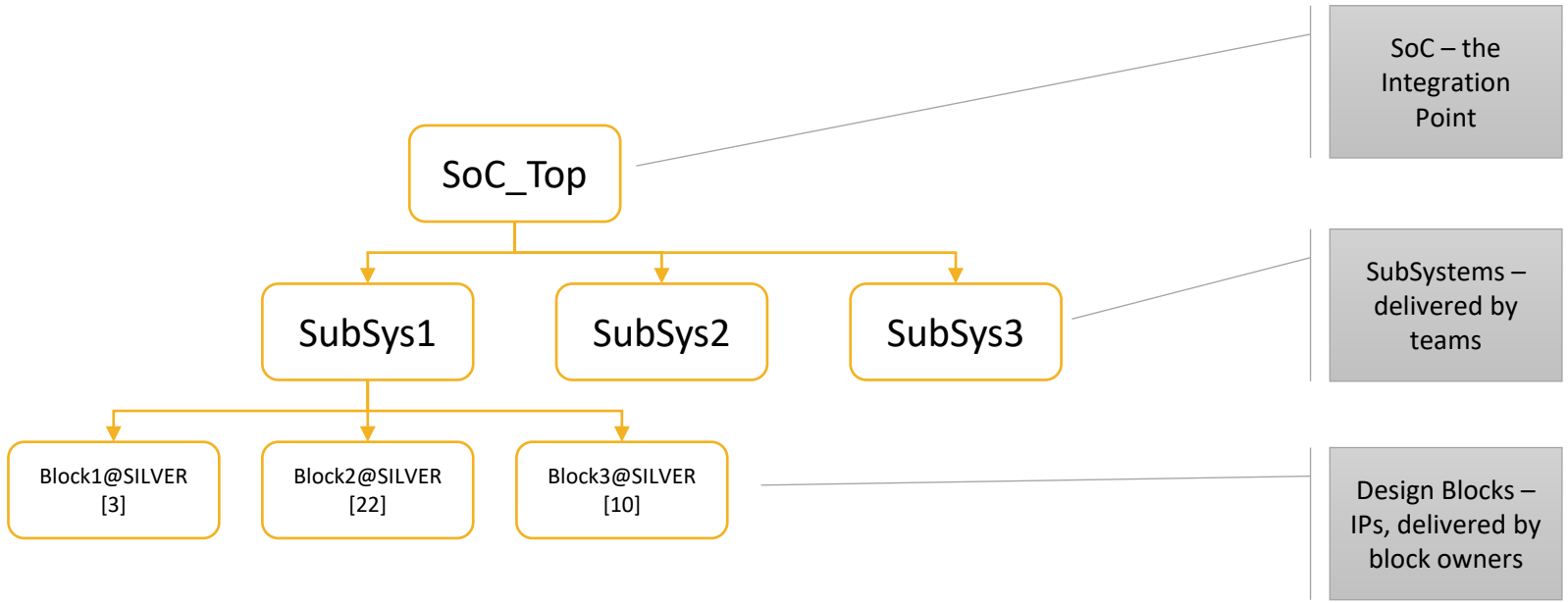
PERFORCE

SoC Bill Of Materials (BoM)

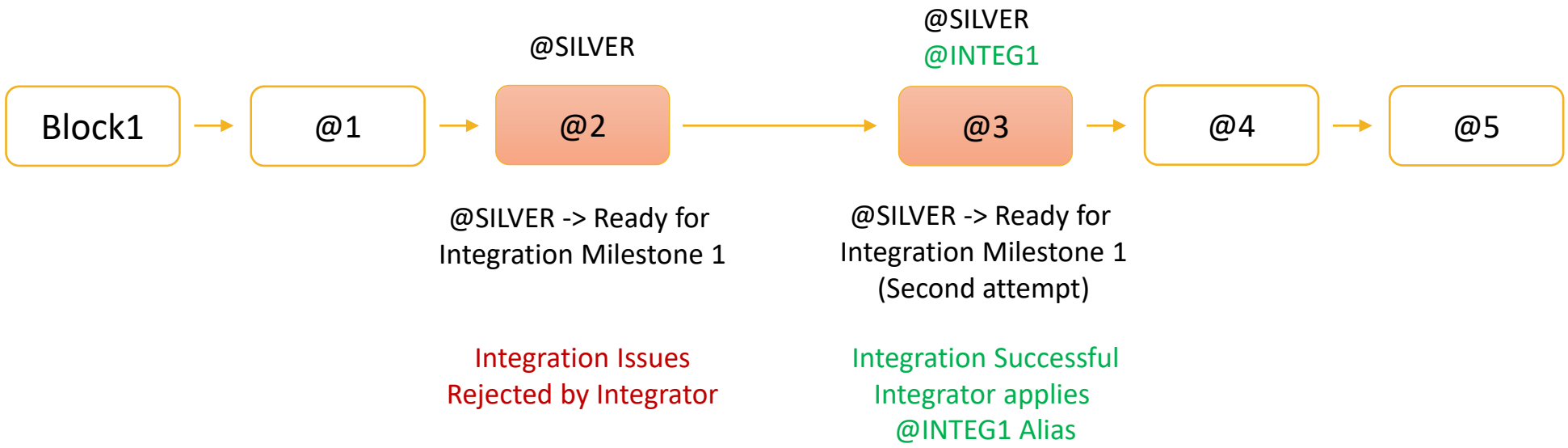
- A BoM in this context is a hierarchical dependency tree of IPs.
- Each IP in the hierarchy can be backed by a different DM (Perforce, Git, SVN, etc).
- Everything is an IP – even the “Project” is just another IP in the system. This allows all IPs to be treated in a similar manner.
- A SW designer might consider the top-level SW IP as their project, while the HW designer might consider the top level HW IP as their project. The Platform designer considers those to as sub-systems.



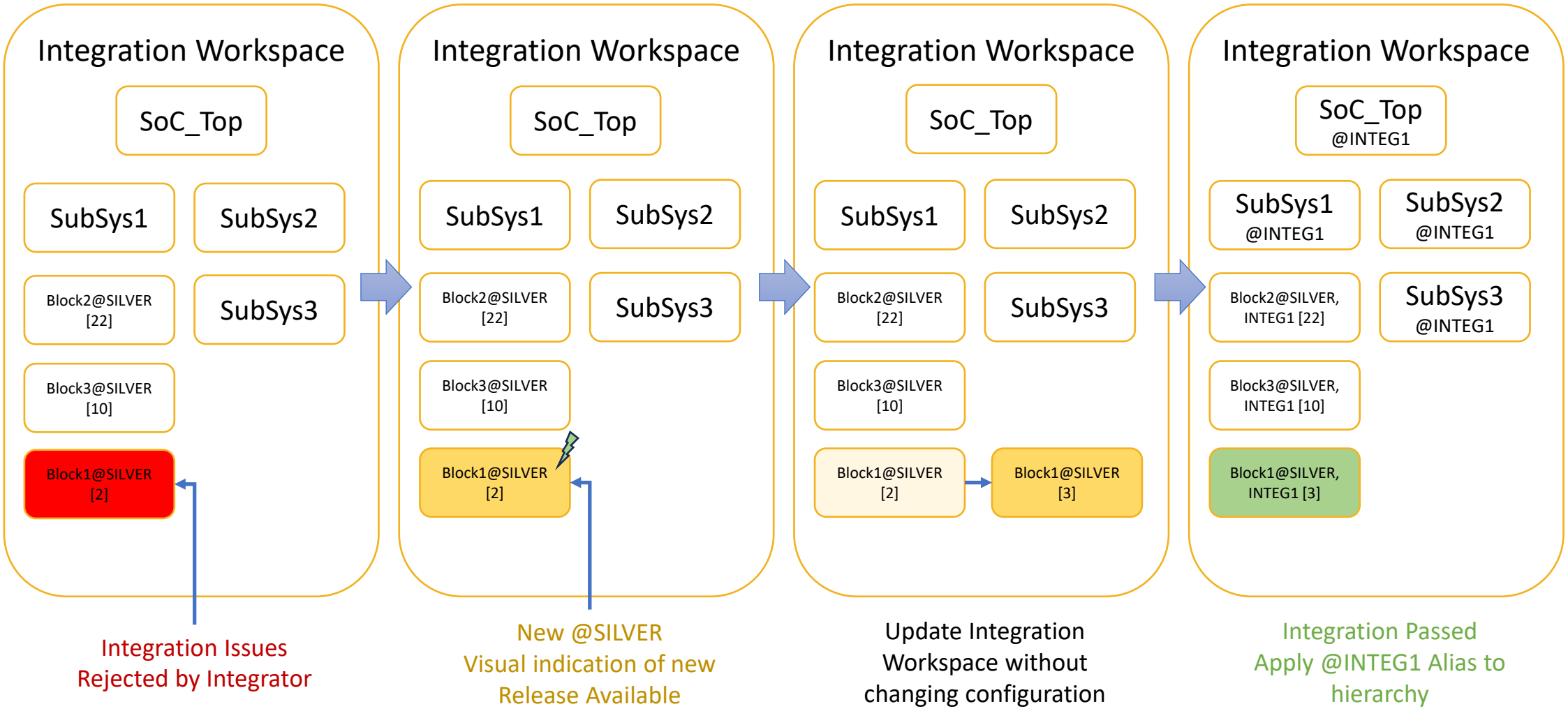
Sample SoC Integration Workflow Using Dynamic Aliases



Integration Lifecycle – IP



Integration Lifecycle – Integrator



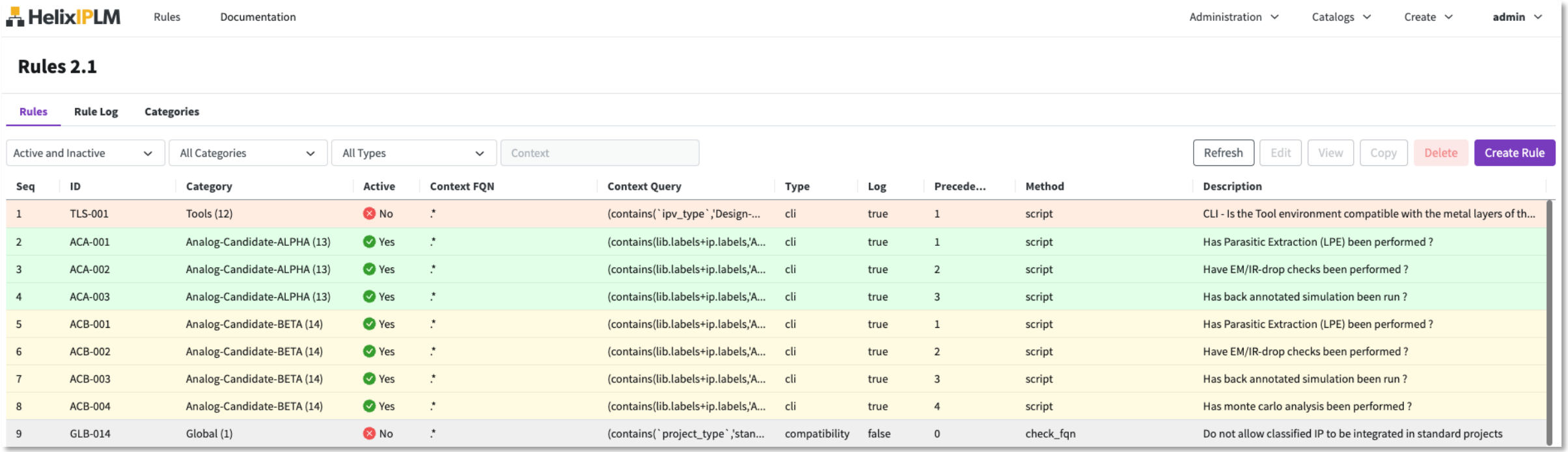
PERFORCE

Rules and Enforcement

Helix IPLM Solution Rules Engine

- Server-side rules (IP add, edit etc)
- Workspace rules (pre_release, post_release etc)
- Managing checklists that can be associated with IP/IPV and workspaces
- Job scheduling for non-blocking rules

Rule: Enforce Silver quality IP versions before integration



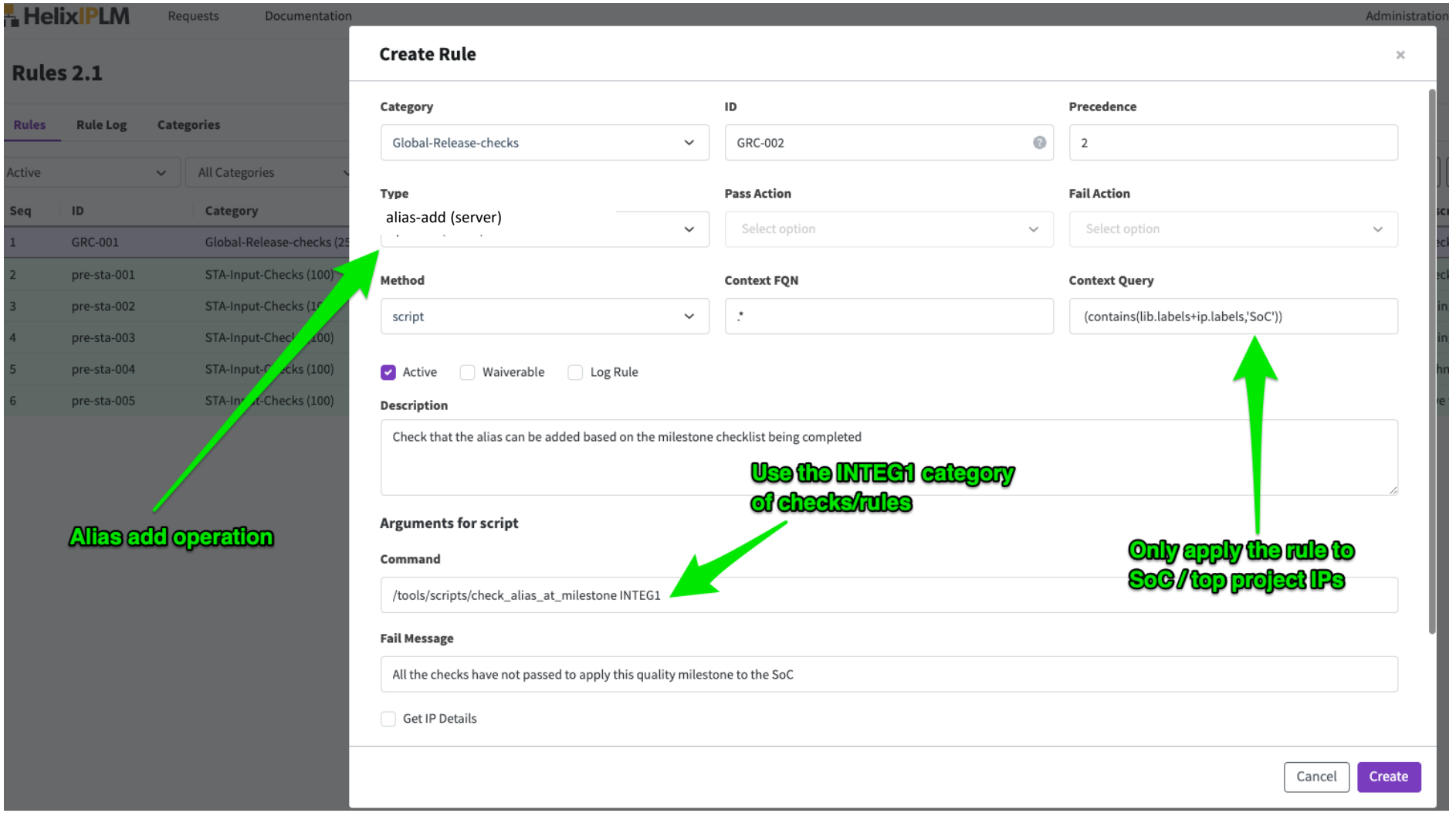
Seq	ID	Category	Active	Context FQN	Context Query	Type	Log	Precede...	Method	Description
1	TLS-001	Tools (12)	No	.*	(contains(`ipv_type`,'Design-...	cli	true	1	script	CLI - Is the Tool environment compatible with the metal layers of th...
2	ACA-001	Analog-Candidate-ALPHA (13)	Yes	.*	(contains(lib.labels+ip.labels,'A...	cli	true	1	script	Has Parasitic Extraction (LPE) been performed ?
3	ACA-002	Analog-Candidate-ALPHA (13)	Yes	.*	(contains(lib.labels+ip.labels,'A...	cli	true	2	script	Have EM/IR-drop checks been performed ?
4	ACA-003	Analog-Candidate-ALPHA (13)	Yes	.*	(contains(lib.labels+ip.labels,'A...	cli	true	3	script	Has back annotated simulation been run ?
5	ACB-001	Analog-Candidate-BETA (14)	Yes	.*	(contains(lib.labels+ip.labels,'A...	cli	true	1	script	Has Parasitic Extraction (LPE) been performed ?
6	ACB-002	Analog-Candidate-BETA (14)	Yes	.*	(contains(lib.labels+ip.labels,'A...	cli	true	2	script	Have EM/IR-drop checks been performed ?
7	ACB-003	Analog-Candidate-BETA (14)	Yes	.*	(contains(lib.labels+ip.labels,'A...	cli	true	3	script	Has back annotated simulation been run ?
8	ACB-004	Analog-Candidate-BETA (14)	Yes	.*	(contains(lib.labels+ip.labels,'A...	cli	true	4	script	Has monte carlo analysis been performed ?
9	GLB-014	Global (1)	No	.*	(contains(`project_type`,'stan...	compatibility	false	0	check_fqn	Do not allow classified IP to be integrated in standard projects

Rules and Enforcement

Other Governance Rules Examples:

- Is the front end and back end compatible / have backend checks been run before release?
- Are we trying to integrate obsolete or retired IP?
- Are we trying to integrate secure or Licensed IP?
- Is the technology properly defined and compatible? (Does it have a PDK as a resource?)
- Has the IP we are integrating at the milestone passed all its checks?
- No Dynamic aliases after a certain maturity
- On promotion/demotion to a lifecycle state are the relevant rules validated?

Integration Rules



PERFORCE

Summary: Automating SoC Integration

Automate integration tasks using IP Governance and Rules

- Automatically pick the right IP Version based on integration task
- For example:
  - 1st Integration Milestone: All IP Versions in the design should be at least “Silver” stage
  - 2nd Integration Milestone: All IP Versions in the design should be at least “Gold” stage

Stage-gate rules prevent bad IP Versions from being integrated

- Reduces the number of integration issues
- Clear and transparent status of blocks that are ready for integration
- Allows integrators to enforce minimum quality requirements across all blocks

Q&A

PERFORCE