Ralph Klos
PROJECTS & CONSULTANCY

GUIDE

Working with the SCP Builder

Pick guardrails, narrow them and attach them to your AWS organization

FOR
Cloud and platform teams

FROM
Ralph Klos Projects & Consultancy

VERSION
SCP Builder 0.4

DATE
6 October 2026

Contents

1. In short . . . 3

2. Opening it . . . 3

3. Working with the builder . . . 4

4. Trusted roles . . . 4

5. Regions . . . 5

6. Enforcing tags . . . 5

7. Starting points per OU . . . 6

8. Importing existing policies . . . 6

9. Own statements . . . 6

10. Other policy types . . . 7

11. Summary for the client . . . 7

12. Limits . . . 8

13. Deploying . . . 8

14. Projects and later changes . . . 9

15. Safety and limits . . . 9

1. In short

The SCP Builder puts together the service control policies (SCPs) most AWS organizations use today. You pick the guardrails, narrow each one to the roles, accounts and regions you want, and get policies that fit AWS's limits, as JSON or Terraform.

- **Nothing to install:** it is one web page you open from your disk.
- **Fully offline:** the page does not call AWS, asks for no keys and sends nothing out. The page fixes that itself in its Content-Security-Policy.
- **You attach them yourself:** the builder makes the policies; you review them and put them in AWS Organizations.
- **Dutch and English:** switch with **NL / EN** at the top. Your choices stay stored in the browser.

> SCPs only restrict: they never grant permissions. They do not apply to the management account or to service-linked roles. What a role's IAM policies allow is capped by the SCP.

2. Opening it

- Unpack or copy the `rkpc-scp` folder to a place of your choice.
- Double-click `index.html`. Any recent browser works: Edge, Chrome, Firefox or Safari.
- The page also works without internet, e.g. on a locked-down admin PC.
- The browser remembers the last state; after a reload the page says which project you continue with. **New project** and **Reset** clear everything, after a confirmation in the page itself.

FILE	CONTENTS
`index.html`	The page
`scp.css`	The house style, light like rkpc.be
`catalog.js`	The guardrails, regions, exemptions and starting points per OU
`others.js`	The other policy types and the StackSet template for backups
`help.js`	The help per environment (the **Help** button)
`app.js`	The scope, the packing into policies, the output, projects

> The **Help** button at the top right explains per situation how to roll out: AWS Organizations, Control Tower, a single account, as code, and changing later.

3. Working with the builder

	STEP	WHAT YOU DO
1	**Trusted roles**	Tick the roles that must always keep working, and add your own roles.
2	**Pick guardrails**	Pick a **starting point per OU** or **Most common**. Then switch on or off what you want; filter by category or search.
3	**Narrow**	Open **Options and scope** per guardrail: settings, exemption, accounts and regions.
4	**Review**	The result on the right shows the number of policies, their size and the slots on the target.
5	**Export**	Pick JSON, Terraform or Other policies and click **Download** or **Copy**. **Summary** at the top gives the overview for the client.

Per guardrail you can narrow the scope like this:

OPTION	WHAT IT PUTS IN THE POLICY
Exempt the trusted roles	`ArnNotLike` on `aws:PrincipalArn`: the roles from step 1 fall outside it.
Only these accounts	`StringEquals` on `aws:PrincipalAccount`.
All except these	`StringNotEquals` on `aws:PrincipalAccount`.
Only in some regions	`StringEquals` on `aws:RequestedRegion`.

> Conditions in one statement apply together. So a guardrail only denies a caller that is not a trusted role, in a chosen account, in a chosen region.

4. Trusted roles

Guardrails skip trusted roles, so Control Tower, StackSets, your pipelines and break-glass access keep working. That also goes for the tagging guardrail: a pipeline role that creates resources without tags is then not blocked. Per guardrail you switch the exemption off.

- **Presets:** Control Tower, StackSets, the Organizations access role and a break-glass role, in one click.
- **Your own roles:** one ARN per line; `*` is allowed, e.g. `arn:aws:iam::*:role/pipeline-*`.
- **Trust by tag:** roles with e.g. `rkpc:trusted = true` are exempt too. Handy when you have many roles you do not want to list by name.
- **The root user** never gets an exemption: that guardrail applies to everyone.

> Trust by tag is only safe if nobody can set the tag themselves. That is why the builder adds the rule `ProtectTrustTag`: only the trusted roles by name may put that tag on, or remove it from, a role, user or session. Without a trusted role by name nobody can set the tag any more; the builder warns about that.

5. Regions

Allow only approved regions denies every call outside the regions you pick. Global services such as IAM, Route 53, CloudFront, billing and support run from us-east-1 and are therefore in an exception list (`NotAction`).

The picked regions are shown as labels at the top; the cross removes one. Add a region by typing its code (the list suggests every commercial region, with its name), then Enter or **Add**, or click a European region under **Quick add**. Several codes with commas work too. A valid code that is not in the list, such as a new region, is accepted and marked "not in the list": check it yourself.

Some services work across regions by design. Tick them if you allow them:

SERVICE	WHY
Bedrock cross-region inference	Inference profiles route a call to another region when the home region is busy.
Amazon Q Developer	Runs from us-east-1.
S3 replication	The replication role writes in the destination region, e.g. for disaster recovery.
AWS Backup copies	A copy job writes into the vault of the destination region.
Amazon Q in chat applications	A global service.

> A ticked service then works in every region, not only for the hand-off. AWS adds global services now and then: compare the list with the current AWS example before you attach it.

6. Enforcing tags

Require tags on new resources blocks new resources without the tags you give. You choose the tag names; allowed values are optional.

SETTING	EFFECT
Tag name, no values	The tag must be there, with any value. `Null` on `aws:RequestTag/<name>`.
Tag name with values	The tag must be there, with one of the values. An extra `StringNotEquals`.
Services	16 services, with **All** or **None** in one click: EC2, RDS and Aurora, CloudFormation, Lambda, ECS, EKS, ECR, DynamoDB, EFS, KMS, Secrets Manager, SNS, SQS, CloudWatch Logs, Step Functions and Kinesis.
Protect	Nobody removes the tags or sets another value later.

Under **Tag policy** you get the matching tag policy. It fixes the spelling and values and shows deviations in the Tag Editor; the SCP does the blocking. Enable tag policies in AWS Organizations first.

> Why not simply "all services"? An SCP can only require a tag on a create call that carries tags (`aws:RequestTag`). A Deny on all actions without a tag would also block every read and list call. The builder therefore covers the services where it can; follow up on the rest with the tag policy and Tag Editor. CloudWatch Logs is off by default: Lambda and other services create log groups without tags. CloudFormation on means CDK and StackSet deploys need tags too.

> Case matters: `CostCenter` is not `costcenter`. Tag an EC2 instance and its volumes in the same launch (`TagSpecifications`), or the guardrail denies the launch.

7. Starting points per OU

Every OU asks for something else. Pick a starting point in the list at the top and click **Apply**: the guardrails and the other policies for that OU are then on. After that you fine-tune as always.

STARTING POINT	FOR
Root	What every account inherits, plus the AI opt-out. Keep it small: every OU below counts this SCP too.
Security	Log archive and audit: as few changes as possible.
Infrastructure	The network account manages internet gateways and VPN; so not those guardrails.
Workloads production	Strict: encryption, central networking, tags, no IAM users; with the EC2 declarative policy and the backup policy.
Workloads test and development	The same baseline with more freedom in networking and a brake on expensive instance types.
Sandbox	Free to experiment inside a fence: regions, cost, AI models, no connections to the outside.
Suspended	Everything closed except for the trusted roles, for accounts being closed.

8. Importing existing policies

With **Import existing policies** the builder reads SCPs that are already attached: one file, a list of policies or the RKPC Toolkit's `findings.json`. Statements the builder recognises become guardrails again, with their settings. Everything else goes word for word into own statements: nothing is lost.

> The toolkit stores every SCP with the OUs and accounts it is attached to. Import `findings.json` and you see straight away what you are building on.

9. Own statements

Something the catalog does not have? Paste it under **Own statements**: one statement, a list or a whole policy. It goes into the same SCPs, the Terraform and the project file. Tick whether the trusted roles are exempt there too. The card shows under every filter.

10. Other policy types

Next to SCPs, AWS Organizations has other policies. You enable each type once at the root (`aws organizations enable-policy-type`); the **Help** button gives the commands.

POLICY	WHAT IT DOES
Resource control policy	A data perimeter on your buckets, keys, queues and secrets: TLS only, your organization only, AWS services only on your behalf.
EC2 declarative policy	AWS fixes the setting itself: no public AMIs or snapshots, IMDSv2, no serial console.
AI opt-out	AWS AI services do not use your content for improvement.
Backup policy	The same backup plan in every account, based on a tag.

The backup policy refers to a vault and a role that must exist in every account. That is why **Other policies** also has a **StackSet template** (CloudFormation) that makes them: the vault in every region, the role once per account. When you deploy it, you fill in the ARN of your KMS key and optionally the organization ID for a central backup account.

```
aws cloudformation create-stack-set --stack-set-name rkpc-backup \
  --template-body file://rkpc-backup-stackset.json --capabilities CAPABILITY_NAMED_IAM \
  --permission-model SERVICE_MANAGED --auto-deployment
Enabled=true,RetainStacksOnAccountRemoval=false \
  --parameters ParameterKey=KmsKeyArn,ParameterValue=arn:aws:kms:eu-west-1:111122223333:key/...
aws cloudformation create-stack-instances --stack-set-name rkpc-backup \
  --deployment-targets OrganizationalUnitIds=ou-xxxx-xxxxxxxx --regions eu-west-1 eu-central-1
```

A KMS key is per region: with one key ARN for every region, use a multi-Region key, or leave the field empty for Backup's AWS managed key. The syntax of these policy types was checked against the AWS documentation on 2 October 2026; check again if that date is long past.

11. Summary for the client

Summary at the top makes an overview with the RKPC letterhead: client, target, date and reference, what is on and how far it reaches, the trusted roles, the files, what is still to resolve and the next steps. The window shows the A4 pages straight away as they come out on paper, with a header and footer line and page numbers; a long table carries on with its column headings at the top of the next page. **Print or PDF** (or Ctrl+P) prints exactly those pages, whatever the margins in the print dialog. Only switch off **Headers and footers** there; the browser proposes the reference as the file name.

12. Limits

- **10,240 characters per SCP.** The builder spreads the statements over as few policies as fit. The meter per policy counts the compact JSON.
- **10 SCPs per target**, FullAWSAccess included. Beyond that the builder warns: attach some at a parent OU, which are inherited.
- **Other types:** an RCP 5,120 characters and 5 per target (RCPFullAWSAccess included); declarative, backup and tag policies 10,000 characters; an AI opt-out policy 2,500 characters. That is what the AWS quotas said on 2 October 2026; AWS raises them now and then, so when in doubt check the page "Quotas and service limits".
- **Readable or compact:** Download gives readable JSON. The console and the Terraform output strip the whitespace themselves. For the CLI or the API, tick **Compact**, because there every space counts.

13. Deploying

Always attach to a sandbox OU first. Watch CloudTrail for `AccessDenied` for a few days and only then move on to test and production.

13.1 In the console

AWS Organizations, **Policies**, **Service control policies**, **Create policy**. Paste the JSON, save, and attach the policy to the OU under **Targets**.

13.2 With the CLI (tick Compact)

From the management account, with your own profile:

```
aws organizations create-policy --type SERVICE_CONTROL_POLICY \
  --name rkpc-guardrails-1 --description "RKPC guardrails" \
  --content file://rkpc-guardrails-1.json
aws organizations attach-policy --policy-id p-xxxxxxxx --target-id ou-xxxx-xxxxxxxx
```

13.3 With Terraform

Pick **Terraform** and click **Download**: you get `rkpc-guardrails.tf` and the JSON files next to it. Put them in one folder and pass the target:

```
terraform init
terraform plan -var target_id=ou-xxxx-xxxxxxxx
```

13.4 Testing in the sandbox OU

- **What must be denied:** in a sandbox account, with an ordinary role, try a few actions a guardrail forbids, e.g. starting something in a region outside your list. The error then names "an explicit deny in a service control policy".
- **What must keep working:** repeat it with a trusted role (break-glass, your pipeline) and run your normal deployment. An `AccessDenied` you did not expect shows up in CloudTrail.
- **Other policy types:** from the management account, look at what an account actually gets:

```
aws organizations describe-effective-policy --policy-type TAG_POLICY --target-id 111122223333
aws organizations describe-effective-policy --policy-type BACKUP_POLICY --target-id 111122223333
```

- The same with `AISERVICES_OPT_OUT_POLICY` and `DECLARATIVE_POLICY_EC2`. Roll out the StackSet before you attach the backup policy, or the vault or the role is missing.

- **Back:** if something gets stuck, detach the policy with `aws organizations detach-policy`; that takes effect at once. Always leave FullAWSAccess attached.

13.5 A single account without Organizations

SCPs only exist in AWS Organizations, and do not apply to the management account. So create a new, empty management account and invite the existing account as a member. For now the JSON also works as an IAM policy on your roles, but that can be detached and does not cover the root user.

14. Projects and later changes

Name the project and click **Save project**. You get a readable JSON file with every setting: guardrails, scope, regions, tags, exemptions and the trust tag. The Terraform download includes it too. Put it in Git next to the policies.

- When the client asks for a change, click **Open project** and pick the file.
- Every screen is back as it was saved. Change what is needed.
- Save the project again and have the difference reviewed (`git diff`) before you attach.

The file also keeps the policies as they were built. If a newer version of the builder makes something different, the page warns when you open it: then compare with what is attached. A file that does not come from the builder is refused; unknown or wrong values are ignored.

> The project file holds the client's account IDs and role names. Keep it where the policies are, not in a public repository.

15. Safety and limits

- **No AWS access:** the builder asks for no keys, calls no AWS API and sends nothing out.
- **Public sources:** the guardrails follow the examples in the AWS Organizations user guide and the AWS Security Reference Architecture.
- **A starting point, not a guarantee:** every organization is different. Test in a sandbox OU and read the policy before you attach it.
- **The management account** never falls under an SCP. Keep no workloads in it.

Questions? Email ralph@rkpc.be or visit rkpc.be.