include::attributes.txt[]

// Attributes :https–github-com-aws-aws-cdk-blob-main-packages-aws-cdk-lib-cx-api-FEATURE-FLAGS-md: https://github.com/aws/aws-cdk/blob/main/packages/aws-cdk-lib/cx-api/FEATURE\_FLAGS.md

[.topic] :info\_titleabbrev: Feature flags :keywords: {aws} CDK, Feature flags

[#featureflags] = {aws} CDK feature flags

== [abstract]

## The {aws} CDK uses *feature flags* to enable potentially breaking behaviors in a release. Flags are stored as xref:context[Context values and the {aws} CDK] values in cdk.json (or ~/.cdk.json). They are not removed by the cdk context --reset or cdk context --clear commands.

// Content start

The {aws} CDK uses *feature flags* to enable potentially breaking behaviors in a release. Flags are stored as xref:context[Context values and the {aws} CDK] values in cdk.json (or ~/.cdk.json). They are not removed by the cdk context --reset or cdk context --clear commands.

Feature flags are disabled by default. Existing projects that do not specify the flag will continue to work as before with later {aws} CDK releases. New projects created using cdk init include flags enabling all features available in the release that created the project. Edit cdk.json to disable any flags for which you prefer the earlier behavior. You can also add flags to enable new behaviors after upgrading the {aws} CDK.

A list of all current feature flags can be found on the {aws} CDK GitHub repository in {https–github-com-aws-aws-cdk-blob-main-packages-aws-cdk-lib-cx-api-FEATURE-FLAGS-md}[FEATURE\_FLAGS.md]. See the CHANGELOG in a given release for a description of any new feature flags added in that release.

[[featureflags-disabling,featureflags-disabling.title]] == Reverting to v1 behavior

In CDK v2, the defaults for some feature flags have been changed with respect to v1. You can set these back to false to revert to specific {aws} CDK v1 behavior. Use the cdk diff command to inspect the changes to your synthesized template to see if any of these flags are needed.

@aws-cdk/core:newStyleStackSynthesis:: Use the new stack synthesis method, which assumes bootstrap resources with well-known names. Requires xref:bootstrapping[modern bootstrapping], but in turn allows CI/CD via xref:cdk-pipeline[CDK Pipelines] and cross-account deployments out of the box.

@aws-cdk/aws-apigateway:usagePlanKeyOrderInsensitiveId:: If your application uses multiple Amazon API Gateway API keys and associates them to usage plans.

@aws-cdk/aws-rds:lowercaseDbIdentifier:: If your application uses Amazon RDS database instance or database clusters, and explicitly specifies the identifier for these.

@aws-cdk/aws-cloudfront:defaultSecurityPolicyTLSv1.2\_2021:: If your application uses the TLS\_V1\_2\_2019 security policy with Amazon CloudFront distributions. CDK v2 uses security policy TLSv1.2\_2021 by default.

@aws-cdk/core:stackRelativeExports:: If your application uses multiple stacks and you refer to resources from one stack in another, this determines whether absolute or relative path is used to construct {aws} CloudFormation exports.

@aws-cdk/aws-lambda:recognizeVersionProps:: If set to false, the CDK includes metadata when detecting whether a Lambda function has changed. This can cause deployment failures when only the metadata has changed, since duplicate versions are not allowed. There is no need to revert this flag if you’ve made at least one change to all Lambda Functions in your application.

The syntax for reverting these flags in cdk.json is shown here.

== [source,json,subs=“verbatim,attributes”]

{ “context”: { “@aws-cdk/core:newStyleStackSynthesis”: false, “@aws-cdk/aws-apigateway:usagePlanKeyOrderInsensitiveId”: false, “@aws-cdk/aws-cloudfront:defaultSecurityPolicyTLSv1.2\_2021”: false, “@aws-cdk/aws-rds:lowercaseDbIdentifier”: false, “@aws-cdk/core:stackRelativeExports”: false, “@aws-cdk/aws-lambda:recognizeVersionProps”: false } } —