Table of JSON Rule Components

|  |  |  |  |
| --- | --- | --- | --- |
| Section | Key | Description | Details |
| ruleGrammarVersion | - | Specifies the version of the rule grammar or schema used. | "5.0" |
| metadata | uid | Unique identifier for the rule. | "0x345345534" |
|  | ruleName | Human-readable name of the rule. | "CustomerEligibilityRule\_v5" |
|  | domain | Domain or category of the rule. | "Eligibility" |
|  | description | Brief description of the rule's purpose. | "Determines customer eligibility (with SageMaker and S3)." |
|  | copiedFromRule | Name of the rule this was copied from. | "EligibilityRule2" |
|  | createdBy | Identifier of the creator. | "corpid" |
|  | approvedBy | Identifier of the approver. | "corpid" |
|  | status | Current status of the rule. | "active" |
|  | copyAllowed | Whether the rule can be copied. | true |
|  | appearsInSearch | Whether the rule appears in search results. | true |
|  | usageScopes | List of scopes where the rule can be used. | Array of objects with  scopeName  and  allowed  (e.g.,  "LoanApprovalWorkflow": true  ,  "InsuranceQuote": false  ,  "CrossSellCampaign": true  ) |
|  | maxScopeAssociations | Maximum number of scope associations. | 1 |
|  | isFinal | Whether the rule is final and uneditable. | false |
|  | allowAllScopes | Whether the rule can be used in all scopes. | false |
|  | ruleExpiry | Expiry information for the rule. | Object with  expiryFlag: "Y"  and  dateOfExpiry: "2025-12-31T23:59:59Z" |
|  | createdDate | Timestamp of creation. | "2024-01-15T10:00:00Z" |
|  | lastUpdated | Timestamp of last update. | "2024-03-03T15:30:00Z" |
|  | ruleVersion | Version of the rule. | "2.0" |
|  | categories | Categories or tags for the rule. | ["Eligibility", "Tasks"] |
|  | dependencies | List of dependent rules. | ["CustomerRule1", "AccountRule1"] |
| cache | cachable | Whether the rule's results can be cached. | true |
|  | ttl | Time-to-live for cached results (seconds). | 3600  (1 hour) |
|  | cacheKey | Template for generating the cache key. | "${ruleName}\_${personaId}" |
| logging | defaultLevel | Default log level. | "INFO" |
|  | defaultDestination | Default log output destination. | "console" |
|  | appendTimestamp | Whether to append timestamps to logs. | true |
|  | format | Format string for log messages. | "[{timestamp}] [{level}] {message}" |
| security | permissions | Access control settings. | Object with  owners  ,  groups  , and  others |
|  | logEncryption | Log encryption settings. | Object with  enabled: true  ,  algorithm: "AES-256"  ,  keyReference: "kms:alias/my-kms-key"  ,  encryptionContext: { "purpose": "logEncryption", "environment": "production" } |
|  | fieldEncryption | Field encryption settings. | Object with  defaultAlgorithm: "AES-256"  ,  defaultKeyReference: "kms:alias/my-kms-key" |
| attributeDefinitions | customer | Defines attributes for the customer entity. | Includes  customer\_id  (string, encrypted),  firstName  (string),  lastName  (string),  email  (string, encrypted, regex pattern),  age  (integer, min: 0) |
|  | account | Defines attributes for the account entity. | Includes  balance  (number),  account\_type  (string) |
| actions | type: "always" | Actions executed regardless of rule outcome. | 1.  postToQueue  (Queue: "eligibleCustomers", retry 3 times on error), 2.  log  (Message: "Customer eligibility rule executed.") |
|  | type: "onSuccess" | Actions executed on successful rule evaluation. | 1.  log  (Message: "Customer eligible. Sending confirmation email."), 2.  conditional  (Condition: "${customer.email != null}", Action: sendEmail, Else: log) |
|  | type: "onFailure" | Actions executed on rule failure. | 1.  log  (Message: "Customer ineligible. Sending rejection notification."), 2.  sendSMS  (To: "$customerPhone", Message: "Your eligibility check was unsuccessful.") |
|  | type: "onError" | Actions executed on execution errors. | 1.  log  (Message: "An error occurred during rule execution."), 2.  notifyAdmin  (Message: "Error in CustomerEligibilityRule") |
|  | type: "afterRun" | Actions executed after rule execution completes. | 1.  log  (Message: "Customer eligibility rule processing complete."), 2.  custom  (Action: "updateCustomerMetrics", Parameters: customerId, ruleName, result) |
| parameters | customerId | Mandatory string parameter for customer identification. | Type:  string  , Mandatory:  true |
|  | accountNumber | Optional string parameter for account number. | Type:  string  , Mandatory:  false |
|  | branchId | Optional string parameter for branch identifier. | Type:  string  , Mandatory:  false  , Optional behavior:  "skipBlock" |
|  | customerStatus | Mandatory string parameter for customer status. | Type:  string  , Mandatory:  true  , Allowed values:  ["active", "inactive", "pending"]  , Default:  "active" |
|  | personaId | Mandatory MID parameter for marketing ID. | Type:  MID  , Mandatory:  true |
|  | expectedFirstName | Mandatory string parameter for expected first name. | Type:  string  , Mandatory:  true |
| preCondition | op: "and" | Operation combining preconditions. | 1.  exists  (Entity:  customer  , Filter:  customer\_id equals $customerId  ), 2.  not  (Subcondition:  or  , Entity:  calendar  , Function:  getCurrentDayOfWeek  , Condition:  dayOfWeek equals "Saturday" or "Sunday"  ) |
| ruleDefinition | op: "and" | Core logic of the rule, combining multiple conditions. | Conditions include customer filters (e.g.,  age >= 18  ), account filters (e.g.,  balance > 1000  ), fraud checks, and dependencies on other rules (e.g.,  CustomerRule1  ) |
| dataExtraction | entityName: "customerDetails" | Specifies data to extract and transform. | Entity:  customerDetails  , Filters:  customer\_id  , Attributes:  ["customer\_id", "firstName", "lastName", "email", "segment"]  , Aggregator:  sum of purchaseAmount  , Transformation:  toUpperCase(firstName) |
| dataContext | dataSources | Defines database sources. | e.g.,  customerDb  (Connection:  customerDbConnection  , Entity:  customer  , Query:  getCustomer  , Attributes:  ["customer\_id", "age", "status"]  ) |
|  | apis | Defines API sources. | e.g.,  accountApi  (Endpoint:  "https://example.com/api/accounts"  , Attributes:  ["customer\_id", "account\_type", "balance"]  ) |
|  | functions | Defines internal functions. | e.g.,  calendar  (Function:  getCurrentDayOfWeek  , Attributes:  ["dayOfWeek"]  ) |
|  | cloudServices | Defines cloud service integrations. | e.g.,  fraudCheck  (AWS Lambda:  "arn:aws:lambda:your-region:your-account-id:function:fraudCheckFunction"  , Attributes:  ["fraudScore", "isFraudulent"]  ) |
|  | graphql | Defines GraphQL sources. | e.g.,  audienceSegments  (Endpoint:  "https://api.example.com/graphql"  , Query:  "query GetAudienceSegment..."  , Mapping:  segmentID, varID, Name  ) |
| result | ruleResult | Defines the structure of the rule's execution result. | Object with  status  ,  timestamp  ,  executionTimeMs  ,  failureReasons |
|  | extractedData | Sample extracted data. | Object with  customerDetails  (e.g.,  firstName: "JOHN", email: "john.doe@gmail.com"  ) |
|  | ruleAttributes | Additional attributes. | Object with  customer  and  account  attributes |
|  | outcome | Defines success and failure outcomes. | Object with  success  (e.g.,  offerId: "OFF123"  ) and  failure  (e.g.,  reasonCode: "CUST\_INELIGIBLE"  ) |

| **Block** | **Key Fields / Sub-Blocks** | **Description / Purpose** |
| --- | --- | --- |
| **metadata** | - **uid** - **ruleName** - **domain** - **description** - **copyAllowed** - **appearsInSearch** - **usageScopes** (array of {scopeName, allowed}) - **maxScopeAssociations** - **allowAllScopes** - **isFinal** - **createdBy**, **approvedBy**, **status** - **ruleExpiry** ({ expiryFlag, dateOfExpiry }) - **createdDate**, **lastUpdated** - **ruleVersion** - **categories**, **dependencies** | High-level information about the rule, including: - **Identity** (e.g., uid, ruleName, domain) - **Lifecycle** (creation date, approval, last update) - **Visibility & Usage** (copyAllowed, appearsInSearch, usageScopes, maxScopeAssociations, allowAllScopes) - **Status & Versioning** (status, ruleVersion, ruleExpiry) - **Categorization** (categories, dependencies) |
| **cache** | - **cachable** - **ttl** - **cacheKey** | Defines caching behavior for the rule engine.  - **cachable**: Whether rule evaluations should be cached. - **ttl**: Time-to-live in seconds. - **cacheKey**: Template or expression forming the unique cache key (e.g., \"${ruleName}\\_${personaId}\"). |
| **logging** | - **defaultLevel** - **defaultDestination** - **appendTimestamp** - **format** | Configures how the rule logs messages.  - **defaultLevel**: e.g., INFO. - **defaultDestination**: e.g., console or file. - **appendTimestamp**: Whether to add timestamps. - **format**: Format string for log messages (e.g., \"[{timestamp}] [{level}] {message}\"). |
| **security** | 1) **permissions** - **owners** (array of {userId, privileges[]}) - **groups** (array of {groupId, privileges[]}) - **others** (object with privileges[]) 2) **logEncryption** 3) **fieldEncryption** | Houses security and access-control configurations: - **permissions**: UNIX-like or custom privileges (view, read, copy, write, execute) for owners, groups, and others. - **logEncryption**: Settings for encrypting logs (algorithm, key reference, etc.). - **fieldEncryption**: Default or field-level algorithms to protect sensitive data (e.g., AES-256). |
| **attributeDefinitions** | - **customer** object - **account** object | Schema definitions for entities used by this rule.  - Each field has a type, description, and whether to encrypt (e.g., customer.age, account.balance). - Includes optional validation constraints (like regex patterns). |
| **actions** | *Array of phases:* - **type: "always"** - **type: "onSuccess"** - **type: "onFailure"** - **type: "onError"** - **type: "afterRun"**  *Each has sub-actions:* e.g., postToQueue, log, conditional, sendEmail, sendSMS, notifyAdmin, etc. | Defines what happens at various stages of rule execution: - **always**: Actions run regardless of pass/fail. - **onSuccess**: Actions if rule condition passes. - **onFailure**: Actions if rule condition fails. - **onError**: Actions if an error/exception occurs. - **afterRun**: Post-execution actions.  Each action can specify error-handling (e.g., strategy: "retry"). |
| **parameters** | Key parameter fields, e.g.: - **customerId** (mandatory) - **accountNumber** (optional) - **customerStatus** (default = "active", allowedValues, etc.) - **personaId**, **expectedFirstName** | Configuration for dynamic inputs that the rule requires.  - **value**: Where the parameter comes from (e.g., $customerId). - **mandatory/optional\*\*: If it must be provided.<br>- \*\*default, \*\*allowedValues\*\*: Constraints or fallback.<br>- \*\*optionalFilterBehavior`**: How to handle absent optional fields (skip filter, treat as no data, etc.). |
| **preCondition** | - Logical **op: "and"** - Terms e.g., exists(customer), and a not condition referencing a function (calendar). | Conditions that must be **true before** the main ruleDefinition is evaluated.  - Example: Customer record must exist, and today isn’t Saturday or Sunday. If preCondition fails, the rule generally won’t proceed. |
| **ruleDefinition** | - **op: "and"/"or"/"not"/"exists"/"ifelse"** - Array of terms with field comparisons, entity lookups, aggregator usage, external function calls, etc. | Main logic that determines if the rule **passes** or **fails**.  - Complex nested operations: and, or, aggregator checks, data entity references (customer, account, orders), array conditions (any, all), date/regex validations, and possible nested rule references.   * **entityType = "data"** – some piece of external data (from a DB, API, etc.) * **entityType = "function"** – a callable function that returns data/attributes (like calendar info) * **entityType = "lambda"** – a specialized function call to a Lambda or other serverless function * **entityType = "rule"** – a nested rule invocation   All the other “types” (e.g., awsLambda, database, api, s3Bucket, etc.) appear under **dataContext** to show how or where that data/service is implemented at a lower level. |
| **dataExtraction** | - Array of extraction steps (e.g., "customerDetails") - **filters**, **extractedAttributes** - **aggregator** (sum, count, etc.) - **transformation** (JUEL expression) | Defines how to retrieve or transform data from underlying entities **after** or **in parallel** to rule checks.  - For instance, aggregating purchase amounts, converting firstName to upper case, etc.  - Useful for providing additional data to the final result or for further logic. |
| **dataContext** | - **dataSources** (database definitions) - **apis** (REST endpoints) - **functions** (function calls) - **cloudServices** (Lambda, SageMaker, S3, etc.) - **graphql** - **connections** (JDBC or other) | Configuration describing **where data comes from** and how to connect.  - **Databases** (via JDBC), **APIs** (REST), **GraphQL**, **cloud services** (Lambdas, SageMaker), etc. - Each source includes endpoints, queries, attributes, and error/fallback handling. |
| **result** | - **ruleResult**: status, timestamp, executionTimeMs, failureReasons - **extractedData**: e.g. customerDetails - **ruleAttributes**: e.g. customer.age - **outcome**: success/failure returnValues | Defines **output** of the rule engine after evaluation.  - **ruleResult**: Overall status (passed, failed, error), timestamps, reasons for failure, etc. - **extractedData** / **ruleAttributes**: Additional data or context gleaned during evaluation. - **outcome**: What to return or do on success/failure (e.g., awarding a discount code, or returning an error code). |

**Notes**

* The **Metadata** block holds general info about the rule lifecycle, scoping, and usage constraints.
* **Logging** and **Security** define cross-cutting concerns that apply across the rule’s entire lifecycle.
* **Parameters**, **preCondition**, and **ruleDefinition** govern **how** the rule evaluates input data to produce a pass/fail outcome.
* **Actions** specify **what** to do once an outcome is determined (success, failure, error).
* **DataExtraction** (optional) and **dataContext** let you **pull** from and **enrich** data sources.
* The **result** block standardizes how final outputs and statuses are reported, including stored or returned data.

**Rule Grammar Overview**

Our rule grammar defines *what a rule is*, *how it’s evaluated*, and *what actions occur* based on the outcome. This JSON-based structure outlines **all the essential parts** of a rule—from its **metadata** and **permissions** to the **conditions** it checks and the **actions** it triggers.

1. **Metadata**
   * **Description & Versioning**: Explains the rule’s purpose and tracks its lifecycle (creation date, last update, version).
   * **Usage & Scope**: Defines where and how a rule can be used, including search visibility, copy permissions, and contexts where it’s valid.
2. **Parameters**
   * **Dynamic Inputs**: Specifies the variables the rule needs (e.g., a customer’s ID or age).
   * **Constraints**: Outlines which parameters are mandatory, optional, or have default values.
3. **Security & Permissions**
   * **Access Control**: Uses a UNIX-like or role-based model to define who can view, edit, or execute the rule.
   * **Data Protection**: Supports field-level encryption and secure logging, ensuring sensitive fields (like customer info) remain protected.
4. **Preconditions & Main Rule Logic**
   * **Preconditions**: Quick checks before running the main logic (e.g., “Only evaluate this rule on weekdays”).
   * **Rule Definition**: The heart of the rule, describing the *logical conditions* (e.g., age >= 18, status == "active") and optional aggregator usage (summing account balances). If the condition is met, the rule passes; otherwise, it fails.
5. **Actions**
   * **Lifecycle Hooks**: Organizes possible actions into stages like “always run,” “onSuccess,” “onFailure,” “onError,” and “afterRun.”
   * **Action Types**: Can log messages, send notifications (email, SMS), call external services, or queue events. Each action can have its own error-handling policy.
6. **Data Extraction & Context**
   * **Data Retrieval**: Details how and from where data is fetched—databases, APIs, cloud services, or local references.
   * **Enrichment**: Allows transformations (e.g., converting names to uppercase) or aggregations (e.g., summing purchase amounts) before or after rule evaluation.
7. **Results**
   * **Outcome Reporting**: Captures the final status (pass/fail/error), timestamps, and any computed outcomes (e.g., discount codes).
   * **Failure Reasons**: Provides an explanation if the rule fails (e.g., “Age requirement not met”).

**Why This Matters**

* **Consistency**: Every rule follows the same template, simplifying how teams manage logic across the organization.
* **Transparency**: Leadership sees at a glance *who created* a rule, *how long it’s valid*, and *where it’s used*.
* **Security & Compliance**: Sensitive fields and logs can be encrypted; access is tightly controlled to protect data.
* **Scalability**: The engine can easily integrate multiple data sources (databases, APIs, cloud functions) and trigger a variety of actions.

| **Block** | **Key Fields / Sub-Blocks** | **Description / Purpose** |
| --- | --- | --- |
| **ruleGrammarVersion** | **6.0** | The version of the rule grammar (schema) itself. Useful for ensuring backward compatibility or indicating new DSL features. |
| **metadata** | - **uid**, **ruleName**, **domain**, **description** - **copyAllowed**, **appearsInSearch** - **usageScopes** (array of {scopeName, allowed}) - **maxScopeAssociations**, **isFinal**, **allowAllScopes** - **ruleExpiry** (with expiryFlag, dateOfExpiry) - **createdDate**, **lastUpdated**, **ruleVersion** - **categories**, **dependencies** | Contains **high-level info** about the rule: unique ID, name, domain, textual description, as well as **usage constraints** (where the rule can be used). Includes version, creation/modification timestamps, expiry settings, and references to other rules (dependencies).   **Examples**:  • usageScopes = contexts or workflows where this rule is (or is not) allowed.  • isFinal = no further modifications. |
| **cache** | - **cachable** (boolean) - **ttl** (time-to-live) - **cacheKey** (template string) | Defines if/how the **rule’s results** are cached. The engine might reuse results for a certain period (**ttl**), keyed by **cacheKey** (e.g., "${ruleName}\_${personaId}"). |
| **documentation** | - **purpose**, **usageNotes**, **technicalNotes** - **businessOwner**, **technicalOwner** | Additional **descriptive info** to help both business and technical stakeholders. Explains *why* and *how* the rule is used, plus identifies owners for any follow-up questions or approvals. |
| **performance** | - **expectedExecutionTimeMs** - **timeoutMs** - **monitoringEnabled** - **alertThresholdMs** | Provides **performance expectations** and operational constraints.   **Examples**:  • expectedExecutionTimeMs = ~250ms.  • timeoutMs = 2000 ms.  • monitoringEnabled = whether performance metrics are tracked.  • alertThresholdMs = triggers an alert if the rule takes longer than 1000ms, for instance. |
| **logging** | - **defaultLevel** (e.g., "INFO") - **defaultDestination** (e.g., "console") - **appendTimestamp**, **format** | Controls how **log messages** are formatted and where they go. For instance, "[{timestamp}] [{level}] {message}". |
| **security** | 1. **permissions**  - owners, groups, others (with assigned privileges like view, read, write…) 2. **logEncryption** (algorithm, key reference) 3. **fieldEncryption** (default algorithm/key) | Defines **access control** (who can do what with the rule) and **encryption** (for both logs and fields).   - **permissions**: Multi-user or multi-group privileges.  - **logEncryption** & **fieldEncryption**: Mechanisms for securing sensitive info. |
| **attributeDefinitions** | - **customer** object (fields: customer\_id, firstName, email, age, etc.) - **account** object (fields: balance, account\_type) | Data **schema** or model for entities the rule uses (e.g., a Customer or Account). Each field has a type, an optional pattern, and a note on whether to encrypt it. This helps standardize how data is validated and protected. |
| **actions** | - **List** of action blocks, each with a type (always, onSuccess, onFailure, onError, afterRun). - Within each block, you have multiple **actions** (e.g., postToQueue, log, conditional, sendEmail, invokeRule, updateFact). | Specifies **what happens** depending on rule outcomes (success/failure/error).   **Examples**:  • type: "always" -> runs every time.  • type: "onSuccess" -> logs and possibly sends a confirmation email.  • invokeRule: calls another rule (e.g., “IntentRule”) after success.  • updateFact: updates a piece of data/fact within the engine context. |
| **parameters** | - **customerId** (mandatory) - **accountNumber** (optional) - **branchId** (optional, skipBlock) - **customerStatus** (with default and allowedValues) - **personaId**, **expectedFirstName** | Describes **external inputs** (e.g., from a request) that the rule needs.  - **mandatory** vs. **optional**.  - **default** (if no value provided).  - **allowedValues** for certain fields. |
| **preCondition** | - **op: "and"** - Terms: e.g. exists on customer, not on calendar function for weekends | Conditions checked **before** main logic. If these fail, the rule won't proceed. Example: Customer must exist, and it can't be Saturday/Sunday. |
| **ruleDefinition** | - **Multiple** logical operations (and, or, exists, etc.) - References to **entityName** and **entityType** (e.g., data, function, lambda, rule) - Aggregators (sum, count) and comparisons (e.g., gte, contains, matches) | The **core logic** that determines pass/fail.  **Example**: A user is eligible if (customer status = active **or** age >= 18 + is VIP) **and** there’s a pending order, etc.  **entityType** can be data (DB or API), lambda (AWS function), or rule (nested rule calls). |
| **dataExtraction** | - **entityName** (e.g., customerDetails) - **extractedAttributes**, optional **aggregator** - **transformation** (JUEL expression) | Defines how to **retrieve or transform** additional data for the final outcome. Example: summing purchaseAmount, transforming firstName to uppercase, etc. This might be used to store or log extra details in the result. |
| **dataContext** | - **dataSources** (databases) - **apis** (REST endpoints) - **functions** - **cloudServices** (Lambda, SageMaker, S3, etc.) - **graphql** - **connections** | Shows how/where the rule engine **obtains data**. Distinguishes DB queries vs. REST APIs vs. GraphQL vs. cloud resources.   **Examples**: a DB alias “customerDb”, an API alias “accountApi”, or a cloud function alias “fraudCheck”. |
| **result** | 1) **ruleResult**:  - status in ["passed","failed","error"]  - timestamp, executionTimeMs, failureReasons 2) **extractedData** 3) **ruleAttributes** 4) **outcome** (success/failure blocks) | Defines **what the engine returns** after evaluation.   **Includes** final status, time stats, and any data extracted along the way.  **Outcome** can have a “success” path (with e.g., discount codes) or a “failure” path (with reason codes). This separation helps in subsequent business logic or UIs. |

**Additional Notes**

1. **documentation & performance** are new sections providing **human-readable notes** and **operational constraints**, reflecting best practices for clarity and monitoring.
2. In **actions**:
   * **invokeRule** calls another rule by name (e.g. "IntentRule") after a success scenario.
   * **updateFact** modifies an existing fact in the engine’s working memory, enabling advanced forward-chaining logic.
3. **entityType** fields in ruleDefinition can be "data", "function", "lambda", or "rule"—each indicating how the engine fetches or evaluates that entity.
4. **Security** covers both **permissions** (who can view/edit/execute) and **encryption** (logs, fields).

This **table** provides a structured reference so readers can easily see **which part** of the JSON does **what**.

|  |  |  |
| --- | --- | --- |
| **Block** | **Key Fields / Sub-Blocks** | **Description / Purpose** |
| **ruleGrammarVersion** | *6.0* | The version of the rule grammar (schema) itself. Useful for ensuring backward compatibility or indicating new DSL features. |
| **metadata** | * uid, ruleName, domain, description * copyAllowed, appearsInSearch * usageScopes (array of {scopeName, allowed}) * maxScopeAssociations, isFinal, allowAllScopes * ruleExpiry (expiryFlag, dateOfExpiry) * createdDate, lastUpdated, ruleVersion * categories, dependencies | Contains high-level info about the rule: unique ID, name, domain, textual description, and usage constraints (where/how it can be used). Includes version, creation/modification timestamps, expiry settings, and references to other rules (dependencies).  Examples:   * *usageScopes* = contexts or workflows where this rule is (or is not) allowed. * *isFinal* = indicates no further modifications allowed if set true. |
| **cache** | * cachable (boolean) * ttl (time-to-live) * cacheKey (template string) | Defines if/how the rule’s results are cached. The engine might reuse results for a certain period (*ttl*), keyed by *cacheKey* (e.g., "${ruleName}\_${personaId}"). |
| **documentation** | * purpose, usageNotes, technicalNotes * businessOwner, technicalOwner | Additional descriptive info for both business and technical stakeholders:   * **purpose**: High-level reason for the rule’s existence. * **usageNotes**: Guidance on when and how to call the rule. * **technicalNotes**: Implementation details. * **businessOwner**, **technicalOwner**: Points of contact. |
| **performance** | * expectedExecutionTimeMs * timeoutMs * monitoringEnabled * alertThresholdMs | Performance-related constraints and metrics:   * *expectedExecutionTimeMs*: Target average runtime. * *timeoutMs*: Hard cap on runtime. * *monitoringEnabled*: Enables performance tracking. * *alertThresholdMs*: Triggers alerts if execution exceeds this. |
| **logging** | * defaultLevel (e.g. "INFO") * defaultDestination (e.g. "console") * appendTimestamp, format | Controls how log messages appear and where they go. For example, format could be "[{timestamp}] [{level}] {message}". |
| **security** | 1. permissions (owners, groups, others) 2. logEncryption (algorithm, key reference) 3. fieldEncryption (default algorithm/key) | Defines both **access control** and **encryption**:   * *permissions*: Who can view/modify/execute the rule. * *logEncryption* & *fieldEncryption*: How logs and data fields are encrypted (e.g., AES-256, KMS keys). |
| **attributeDefinitions** | * customer object (fields like customer\_id, firstName, email, etc.) * account object (fields like balance, account\_type) | Schema definitions for data entities this rule works with.   * Each attribute has a *type*, *description*, and *encrypt* boolean. * Optional validation (e.g., *pattern* for email). |
| **actions** | * Array of blocks: always, onSuccess, onFailure, onError, afterRun * Inside each block: multiple actions (e.g., postToQueue, log, conditional, sendEmail, invokeRule, updateFact) | Defines **what happens** depending on rule outcomes:   * *always* runs unconditionally. * *onSuccess* triggers if the rule passes (e.g., send an email). * *onFailure* triggers if the rule fails (e.g., send an SMS or log a message). * *onError* triggers on exceptions. * *afterRun* executes post-rule finalization steps.   Actions can also have onError strategies (e.g., retry, fallback). |
| **parameters** | * customerId (mandatory) * accountNumber (optional) * branchId (optional with skipBlock behavior) * customerStatus (has default, allowedValues) * personaId, expectedFirstName | \*\*Dynamic inputs\*\* the rule engine expects:   * Which ones are mandatory vs. optional. * Default values or enum constraints (like ["active","inactive","pending"]). * Optional filter behavior (e.g., "skipBlock" if a parameter is missing). |
| **preCondition** | * op: "and" * Terms checking exists(customer), not (calendar=weekend), etc. | Conditions that must be satisfied *before* the ruleDefinition.   * If *preCondition* fails (e.g., customer doesn’t exist, or it’s a weekend), the rule generally won’t run the main logic. |
| **ruleDefinition** | * op: "and"/"or"/"exists"/"not" logic * Filters comparing fields (customer.age >= 18) * entity references (entityName = "account", etc.) * Aggregators, JUEL expressions | The \*\*core logic\*\* for pass/fail.   * Nested conditions combining *and/or*, aggregator checks, references to other rules or lambda functions, etc. * *entityType* can be "data", "function", "lambda", "rule". * Example: Customer is "active" or older than 18, has pending orders, valid email, etc. |
| **dataExtraction** | * entityName (e.g., "customerDetails") * extractedAttributes, aggregator, transformation | Defines how to \*\*retrieve and transform\*\* additional data, e.g.:   * *aggregator*: Summation of purchase amounts. * *transformation*: Convert firstName to uppercase. * Stored into *extractedData* for further use or final output. |
| **dataContext** | * dataSources, apis, functions * cloudServices, graphql * connections (JDBC, etc.) | Describes \*\*where data is fetched from\*\*:   * Databases (type: "database"), REST APIs, GraphQL, or cloud endpoints (AWS Lambda, SageMaker, S3). * Each includes references to queries, endpoints, or connection details. * *onError* sub-blocks handle fallback or logging if a data call fails. |
| **result** | * ruleResult (status, timestamp, failureReasons) * extractedData * ruleAttributes * outcome (success/failure returnValues) | Defines \*\*what is produced\*\* after the rule runs:   * *ruleResult*: pass/fail/error, execution time, reasons for failure. * *extractedData* & *ruleAttributes*: Additional data captured during evaluation (e.g., customerDetails). * *outcome*: Key/value results for success/failure flows (like discount codes or rejection messages). |