Application Design

ServiceNow Certified Application Developer

KNOWLEDGE CHEATSHEET

FROM NEED TO APPLICATION

Requirements People & process → **Model** Tables & fields → **Scope** App boundary → **Experience** Forms & navigation

01 Designing and creating an application

Requirements and platform fit

- Define the **business problem**, users and stakeholders, process steps, and data inputs and outputs before building. Translate requirements into tables, experiences, roles and automation.
- ServiceNow suits **record-based business processes**, forms, approvals, reporting and integrations. Requirements for unrestricted native libraries or specialized multimedia processing need a separate technical fit assessment.
- **Guided Application Creator** guides initial app setup, data and experiences. Tables can be created or reused, and spreadsheet data can seed a model. **Studio** organizes ongoing application development.

Scope and application files

- An **application scope** groups resources under a namespace and controls interaction with other applications. Its prefix reduces naming collisions; choose the scope when creating the application.
- The **Application Picker** selects the application being edited and sets the development scope. Verify it before creating artifacts; a menu used to run an app serves a different purpose.
- **Application files** are metadata artifacts such as tables, scripts, reports and security rules. Review the application's **Application Files** related list to inspect what is packaged.
- **Scoped applications** provide isolation and a managed installation lifecycle. Scope boundaries govern application interaction; roles and ACLs govern user access. Neither boundary replaces the other.

Tables, fields and inheritance

- A **table** stores records; dictionary definitions describe fields. A record's **sys_id** is its unique identifier. A **schema map** visualizes table relationships, including references and extensions.
- An **extended table** inherits parent fields and applicable behavior; inherited ACLs participate in access evaluation. **Task** is a common parent for work-oriented records. Extension does not copy all parent records into the child.
- Add child-specific fields where needed. Do not delete an inherited field from the child; changes to a parent's field can affect descendants. A child-specific **label** can present an inherited field differently.
- Use **String** for free text, **Choice** for controlled options, **Reference** to link another table, **Date** for a calendar date, and **Date/Time** when time of day matters. Field creation and list layout are separate configurations.

User Interface

ServiceNow Certified Application Developer

KNOWLEDGE CHEATSHEET

THE USER INTERACTION LAYER

- **Modules** — Navigate
- **Forms** — View & edit
- **Client logic** — Guide input
- **Producers** — Capture requests

02 Application user interface

Forms, lists and navigation

- In **Form Designer**, drag existing fields from the field navigator onto the layout; drag a field type to define a new field. Removing a field from a form layout does not delete its dictionary definition.
- Use **sections** to group form fields, **annotations** for instructions, and **related lists** for associated records. Configure list columns separately; adding a table field does not automatically select it for every list view.
- An **application menu** groups **modules**. A module opens a destination such as a list, new-record form, content page or URL; it need not reference a table. One application can have multiple menus.
- A module's **Link type** determines its destination. Use **List of Records** for a filtered list, **New Record** for data entry, and **URL (from Arguments)** for a supplied URL, including suitable producer or properties pages.
- **UI Actions** add commands such as form buttons, form context-menu entries, list banner buttons and list choices. Their scripts may run on the client, server, or use a coordinated client/server pattern.

Client scripts and UI policies

- **Client Scripts** react to browser events: **onLoad**, **onChange**, **onSubmit** and **onCellEdit**. Use client logic for responsive feedback; enforce data integrity and access on the server.
- **g_form** is GlideForm. Common methods include **getValue()**, **setValue()**, **setMandatory()**, **setReadOnly()** and **setDisplay()**. Choose behavior appropriate to the field and the user experience.
- Use **g_form.addInfoMessage()** for a form-level message and **g_form.showFieldMsg(field, message, type)** for field-level feedback. Use browser developer tools and supported field diagnostics to trace client behavior.
- A **UI Policy** declaratively controls mandatory, visible and read-only states. Prefer actions over scripts for simple rules; actions run before policy scripts. **Reverse if false** restores the opposite action state when the condition fails.
- **g_user** supplies client-side user information. **hasRole()** also treats admin as satisfying a role check; **hasRoleExactly()** checks the specified role without that shortcut. A browser role check is not an authorization boundary.

User Interface / Security

ServiceNow Certified Application Developer

KNOWLEDGE CHEATSHEET

SEPARATE ACCESS BOUNDARIES

- **Roles** — User capabilities
- **ACLs** — Record & field access
- **Scope** — Application access
- **Caller controls** — Resource calls

Record producers

- A **Record Producer** presents catalog-style questions and creates a record in a target table. Variables collect user input; they are distinct from target-table fields. Use the catalog request process for requested items.
- Use **Catalog Client Scripts** and **Catalog UI Policies** for producer variables. Map suitable variables to fields; in a producer's server script, read **producer.variable_name** and assign target fields through **current.field_name**.
- A producer can use **producer.redirect** for navigation after submission in the standard interface; portal navigation has its own context. Handle an intentional abort with useful feedback and test that no unintended record is created.

03 Security and restricting access

Roles and access controls

- Use separate **application user** and **application administrator** roles for normal work and configuration. Menu and module roles control navigation visibility; **ACLs** enforce access to records and fields.
- Modules normally respect the parent menu's roles as well as their own. **Override application menu roles** allows the module's role requirement to take precedence. An empty module roles field does not bypass its parent menu or data ACLs.
- Core ACL operations are **create**, **read**, **write** and **delete**. **table.None** addresses record access, **table.field** a named field, and **table.*** a wildcard field rule. A sys_id is not an ACL field selector.
- Within an ACL, required **roles**, **condition** and **script** must pass; any one listed role satisfies the role portion. Evaluation proceeds through roles, condition and script. Empty portions impose no additional test.
- Record access and field access are separate gates. Applicable table and field rules, including inherited and wildcard rules, determine access. Multiple ACLs at the same matching level can provide alternative grants; do not assume every ACL must pass.
- Use **Debug Security Rules** and impersonation to inspect actual ACL decisions. Server ACL scripts can use **gs.hasRole()** and **current.isNewRecord()**; client objects such as g_user are unavailable there.

Application access boundaries

- A table's **Accessible from** setting limits access to its own scope or permits other scopes. Cross-scope **Can read**, **Can create**, **Can update** and **Can delete** settings govern permitted data operations.
- **Can read** is the prerequisite for exposing the other cross-scope CRUD permissions. Application Access uses **update**, while an ACL uses the **write** operation name. Access settings do not grant every user permission.

Security / Automation

ServiceNow Certified Application Developer

KNOWLEDGE CHEATSHEET

SEPARATE ACCESS BOUNDARIES

- **Roles** — User capabilities
- **ACLs** — Record & field access
- **Scope** — Application access
- **Caller controls** — Resource calls

- **Allow configuration** permits supported configuration artifacts from other scopes, such as Business Rules. It does not grant unrestricted data changes; a rule's execution still respects applicable scope restrictions.
- **Allow access to this table via web services** exposes supported service access. The authenticated caller must still have the required permissions and pass applicable ACLs; enabling the option is not anonymous access.

Cross-scope calls and delegation

- **Cross-scope privileges** authorize interaction with another application's resources. Under runtime access **Tracking**, calls are recorded; **Enforcing** requires an allowed privilege before a restricted call can proceed.
- **Restricted Caller Access** adds caller-level controls for supported resources. **Caller Tracking** records callers; **Caller Restriction** requires approval. Its records are distinct from ordinary user ACLs and cross-scope privileges.
- **Delegated Development** lets administrators grant selected developers access to a scoped app and chosen artifact types. Script editing and security configuration can be delegated separately; use least privilege instead of broad admin access.

04 Application automation

Server scripting and Business Rules

- **Business Rules**, **Script Includes**, **Script Actions** and **Scheduled Script Executions** run on the server. **gs** is GlideSystem; **gs.getUserID()** returns the executing user's sys_id and **gs.hasRole()** tests a role.
- A **before Business Rule** validates or changes the current record before database persistence. An **after rule** handles related work after the operation; **async** rules defer server work. Avoid **current.update()** recursion inside rules.
- A **display Business Rule** prepares data before the form is sent to the browser. Set **g_scratchpad** properties to pass server information to client scripts. Do not place secrets in data delivered to a client.
- **current** represents the record in a record-bound context; **previous** holds its earlier state where available. Do not assume previous exists for display or async rules, or that a scheduled job has a current record.
- Business Rule **Actions** can set values or add a message without a custom script. The standard function wrapper is an immediately invoked function expression, which keeps local variables inside the rule.

Automation

ServiceNow Certified Application Developer

KNOWLEDGE CHEATSHEET

CHOOSE THE EXECUTION CONTEXT

- **Business Rules** — Record operations
- **Flows** — Process orchestration
- **Events** — Deferred responses
- **Schedules** — Recurring work

Reusable logic, data and debugging

- Use **Script Includes** for reusable server logic called on demand. A class-based utility uses a constructor/prototype pattern; its methods can serve multiple callers and centralize maintenance.
- **GlideAjax** calls a client-callable Script Include from the browser. Prefer asynchronous callbacks and return only needed data; validate input and enforce permissions on the server.
- **GlideRecord** accesses table records: add filters with **addQuery()**, run **query()**, iterate with **next()**, or retrieve a record with **get()**. Avoid unbounded queries; use **GlideRecordSecure** when ACL enforcement is required.
- Use **GlideDateTime** for server date/time work. Use scoped **gs.info()**, **gs.warn()**, **gs.error()** and **gs.debug()** for logging; debug output depends on logging configuration. Client scripts cannot call gs directly.
- Store configurable settings in **Application Properties** and read them server-side with **gs.getProperty(name, defaultValue)**. A properties page lets authorized administrators change behavior without editing business logic.

Flows, schedules and reports

- **Flow Designer** combines a trigger with actions and flow logic. Common triggers include record changes, schedules and application-specific events. Catalog and integration triggers depend on the relevant application and configuration.
- An **action** performs a reusable operation; a **subflow** packages reusable logic with inputs and outputs and is called by another process. A **spoke** groups reusable integration actions in a scoped application.
- Use **data pills** to pass outputs between steps. Match variable types to the data, such as String, Integer, record references and supported arrays. Review **Run as** permissions and execution details when testing.
- A legacy **workflow context** is a running instance created from a workflow version; it executes activities and follows transitions. A workflow definition, checked-out edit and runtime context are different objects.
- Use a **Scheduled Script Execution** for recurring server work such as cleanup, reminders or database checks. Use **reports** for authorized on-demand analysis or scheduled distribution; sharing a report does not override data security.

Automation / External Data

ServiceNow Certified Application Developer

KNOWLEDGE CHEATSHEET

MOVE DATA DELIBERATELY

- **Import Set** — Stage data
- **Transform Map** — Map & coalesce
- **Web services** — Connect systems
- **Inbound email** — Create or update

Events and email notifications

- The **Event Registry** defines known events; the **Event Log** records event occurrences. **gs.eventQueue(name, record, parm1, parm2)** queues an event that can drive a **Script Action** or **Email Notification**.
- An **Email Notification** defines **when to send**, **who receives** and **what it contains**. Use record substitutions for field content and **${event.parm1}** or **${event.parm2}** for event parameters.
- For competing notifications on the same record with the same recipients, the highest nonzero **weight** wins. Weight **0** is always eligible when sending criteria are satisfied; it does not disable the notification.
- To troubleshoot delivery, check the event, notification conditions and recipients, then the **Email logs**. Verify the recipient is active, has a valid email address and has notifications enabled; distinguish generated email from successful delivery.

05 Working with external data

Imports and transform maps

- A **Data Source** identifies incoming data, such as a spreadsheet, CSV or supported remote source. An **Import Set** stages it; a **Transform Map** maps source columns into fields on the target table.
- A **coalesce** field identifies matching target records. A match updates the target; no match normally inserts a record. With multiple coalesce fields, their combined values must match.
- Choose stable, sufficiently unique coalesce keys and test empty values and duplicates. Coalescing decides insert versus update; it does not automatically cleanse data or merge all pre-existing duplicates.

Web services and API tools

- With **outbound** web services, ServiceNow is the consumer calling another provider. With **inbound** APIs, external clients consume ServiceNow services. Network access, authentication and authorization are separate requirements.
- A **REST Message** defines an outbound call; its **endpoint** is the resource URI. **GET** reads, **POST** creates or submits, **PUT** replaces, **PATCH** partially updates and **DELETE** removes supported resources.
- Configure credentials, headers, parameters and request bodies deliberately. REST and SOAP message templates can use **${variable_name}** placeholders populated at runtime; a **SOAP WSDL** describes the service contract.

External Data / Application Management

ServiceNow Certified Application Developer

KNOWLEDGE CHEATSHEET

APPLICATION DELIVERY LIFECYCLE

Git Version metadata → **ATF** Validate behavior → **Package** Review artifacts → **Distribute** Install & verify

- **REST API Explorer** discovers and tests ServiceNow API endpoints and generates sample request code. Calls use the executing user's permissions; mutating requests can change real instance data.
- A **MID Server** can bridge supported integrations to private networks, including supported source-control connectivity. Credentials alone do not make a repository or API behind a firewall reachable.

Inbound email processing

- **Inbound Email Actions** classify and process incoming messages, creating or updating records. Their scripts can use **email** for message data and **current** for the target record.
- A **watermark** helps match a reply to an existing record. Record numbers in subjects and reply/forward classification can also affect processing; verify matching, conditions and action order when troubleshooting.

06 Managing applications

Source control and Studio

- Link an app to **Git** using a reachable repository URL and an authorized credential. Studio operations include committing changes, applying remote changes, creating branches and stashing local work; use appropriate source-control roles.
- A **branch** separates a line of development, a **tag** names a repository revision, and a **stash** saves local changes temporarily for later reapplication. Source control versions application metadata, not ordinary transactional records.
- Use **Studio code search** to find logic across application artifacts. Code-search groups define the tables searched; configure an appropriate custom group when additional supported file types need inclusion.

Packaging and distribution

- The **Application Repository** distributes custom application versions to an organization's instances. The **ServiceNow Store** offers certified applications and integrations; **Share** provides community-contributed resources.
- An **update set** groups captured configuration changes for transfer between instances. **Customer Update [sys_update_xml]** records hold captured customizations; **Update Set [sys_update_set]** identifies the container.

Application Management

ServiceNow Certified Application Developer

KNOWLEDGE CHEATSHEET

APPLICATION DELIVERY LIFECYCLE

Git — Version metadata → **ATF** — Validate behavior → **Package** — Review artifacts → **Distribute** — Install & verify

- Develop in an **In progress** set, mark it **Complete**, then retrieve, preview, resolve conflicts and commit on the target. Use a new set for later changes to preserve a clear deployment sequence.
- Ordinary table records are not automatically shipped because the table is packaged. Deliberately include required seed/configuration data as **application files** where supported, and verify dependencies before distribution.
- A Script Include's **Read-only** protection permits viewing but prevents editing in the distributed app. **Protected** hides the implementation in applicable Store distribution; test protection in the consuming instance.

Automated testing

- The **Automated Test Framework (ATF)** organizes steps into tests and suites. Cover record operations, form behavior, permissions and supported custom UI components. Run tests in a nonproduction environment.
- Use **Create a User** to prepare test roles and groups, then **Impersonate** for the required identity. Validate behavior as normal users as well as administrators; broad admin privileges can hide missing access controls.
- Review assertions and execution results after changes. ATF can clean up supported test data, but external effects such as outbound integrations need deliberate isolation; do not assume every test operation is reversible.