ChatGPT

Company-Specific Stratus Implementation Guide

Figure: The Stratus Admin dependency hierarchy. Each circle represents a configuration node in the Stratus Admin console, with arrows indicating setup order and dependencies.

This guide walks through each Stratus Admin node (from **Company Settings** down to deliverable definitions) in the required setup sequence. For each node, we indicate whether it is **REQUIRED** or **OPTIONAL**, describe its purpose, show its setup order, and provide company-specific examples from the user's eVolve/Revit data (families, parameters, spool/package IDs, etc.). Tables of field mappings, parameter aliases, and filters illustrate how the user's content (families, shared parameters, points, fittings, etc.) aligns with Stratus fields for **BOM**, **Cut List**, **Assembly**, and **Prefab/Package** deliverables.

Company Settings (REQUIRED, setup order: 1)

What it does: Global configuration for the company. This includes defining company info, user/role assignments, naming conventions, part templates, part aliases, fields, filters, and integration keys (App Keys). These settings apply across all projects and deliverables.

How to configure: In Stratus Admin, go to **Admin > Company**. Key sections include:
- **App Keys:** Define API/partner keys (e.g. for PypeServer integration) under *Company* > *App Keys*. For example, PypeServer Connect reads the Stratus app key under *Admin* > *Company* > *App Keys* to upload prefab packages [1].
- **Project Roles & Stations:** (Optional) Assign roles and default stations if needed.
- **Package Categories:** (Optional) Classify types of prefab packages (e.g. "Building", "Rack").

Example (Company info):
| Field | Example Value | |---|---| | Company Name | *Acme Electric Co.* | | Default Units | *Imperial (inches/feet)* | | Authoring Software | *EVOLVE Electrical* |

Aliases (Part Aliases): Under *Company* > *Part Aliases*, map legacy or vendor names to canonical part names. For example, if the user's content has a part called "Conduit 1/2 EMT", alias it to "Rigid Conduit 1/2"". (Aliasing ensures the Stratus part library matches the user's naming.)

Project Settings (OPTIONAL, setup order: 2)

What it does: Project-specific overrides or defaults. Often used to configure project-level filters, user assignments, or deliverable defaults. Not strictly required to start processing, but can tailor Stratus behavior per project.

How to configure: In Stratus Admin, select **Admin > Project** (after selecting a company and project). Key settings include: *Sync Schedules*, *Default Drawing Sets*, or *Assembly Conflict Resolution*. For example, the *Project*

> *Settings* page lets you specify how assemblies are numbered within that project. (This is optional; if not used, company defaults apply.)

Example: If all projects at the company use a standard prefix (e.g. “ACME-#”), you could set this in Project Settings.

Naming & Numbering (REQUIRED, setup order: 3)

What it does: Defines part numbering rules and naming conventions for spools, assemblies, and packages. This controls how Stratus auto-generates IDs for deliverables.

How to configure: Under *Admin* > *Company* > *Naming and Numbering*, set up naming formats and incrementors:
- **Naming Conventions:** Create patterns (e.g. `SP-{ProjectCode}-{0000}` for spools, `PKG-{ProjectCode}-{000}` for packages) and define default prefixes/incrementors. For example, you might use an **Incrementor** with 5 digits (e.g. `00001`) as noted in the Stratus update logs [2].
- **Default Values:** Set default values (e.g. default starting number, fixed prefixes like project code, and “Authoring Software” tag). For example, add a rule that sets the software to “EVOLVE” under *Admin* > *Company* > *Naming and Numbering* > *Authoring Software* [3].

Example rule: Prefix for spool IDs: `#-SP-`, with a 4-digit incrementor. So the first spool ID might be **“ACME-SP-0001”**. Package IDs might use `PKG-` prefix (e.g. **“ACME-PKG-001”**).

Part Templates (REQUIRED, setup order: 4)

What it does: Defines classes of parts (e.g. “Rigid Conduit”, “Cable Tray”, “Panelboard”) and how their properties map to Stratus fields (BOM and cut list mappings). Each template groups similar Revit families into a Stratus part type.

How to configure: Under *Admin* > *Company* > *Part Templates*, create a template for each major part type. For each Part Template, the UI has tabs for **Properties**, **Cut List Mappings**, and **BOM Mappings** [4].
- **Properties:** Set basic part attributes (unit, description rules).
- **Cut List Mappings:** Map part dimensions for cut lists (e.g. Revit’s *Length* parameter → Stratus *Length* field).
- **BOM Mappings:** Map part parameters for the BOM (e.g. Revit *Type Name* or *Material* → Stratus *Description/Material*).

When a single template is selected, all tabs (Properties, Cut List, BOM, etc.) are active [4]. Edits to fields in these tabs define how Stratus will generate BOM and cut lists for that part type.

Company-specific example: Suppose the company uses EVOLVE families for conduits and panels. Example Part Templates:

Part Template Name	Example Family	Key Mapped Fields
Rigid Conduit	*eE_CF_EMT_Connector* (conduit couplings) [5]	Stratus fields: *Nominal Size* ← Revit *Diameter*, *Material* ← Revit *Material*, *Quantity* (auto)
Conduit Bend	*eE_CF_Bend* family	*Nominal Size* ← *Diameter*, *Angle* ← *Type Name*, *Length* ← *Radius*
Cable Tray (100W)*	Custom *Cable Tray* family	*Width* ← *Type Width*, *Depth* ← *Type Depth*, *Material*, *Finish*
Panelboard	*Panelboard-Family*	*Voltage* ← *eP_Voltage* (shared param), *Catalog Number* ← *Type Name*
Strut Channel	*Strut-Channel*	*Size* ← *Type Name*, *Length* ← *Length*, *Material*
…	…	…

(*Names are illustrative; adjust to actual family/type names in the user's library.)

Fields (REQUIRED, setup order: 5)

What it does: Defines custom fields (Stratus properties) that parts or deliverables will use. Stratus provides standard fields (Length, Size, Material, etc.) and allows adding user-defined fields. For example, fields like *Level ID*, *Custom ID*, or *Spool Number* can be defined here.

How to configure: Under *Admin* > *Company* > *Fields*, review existing fields and add any missing ones to match the user's data needs. For each field, specify data type (text, number, etc.) and default value if needed. These fields can then be mapped from Revit parameters in **Import Mappings** or used in templates.

Example (user's Revit/eVolve fields): The company's Revit models use EVOLVE point parameters and tags (e.g. *eVolve_LevelId*, *eVolve_CustomId*, *eVolve_AreaTypeId*, *eVolve_EstimateId*, *eVolve_KitId* [6]). In Fields, ensure Stratus has corresponding fields such as *Level ID*, *Custom ID*, *Area Type ID*, etc. For instance, define a Stratus field "Custom ID" and map it to the Revit parameter **eVolve_CustomId**.

Filters (REQUIRED, setup order: 6)

What it does: Defines rules to select which Revit elements become Stratus parts or assemblies. Filters can include category checks, parameter conditions, etc. For example, only include conduit of a certain type or only panelboards on certain levels.

How to configure: In *Admin* > *Company* > *Filters*, create filters that match the company's workflows. Each filter targets a part template or deliverable type. For example:
- Filter by **Family Name** or **Category** (e.g. Family = *eE_CF_Connector* for conduit couplings).
- Filter by **Parameter** (e.g. *eP_IsAssembly* = *Yes* to include only spooled elements).
- Filter by **Model Group or Location** (e.g. only include elements in "Prefabrication" model).

Example filters:
| Filter Name | Condition | Applies To | |---|---|---| | ConduitSegments | Family = “*Conduit*” category | Rigid Conduit (template) | | StrutMaterials | Material = “Galvanized Steel” | Strut Channel (template) | | Panelboards | Category = Electrical Equipment & Type = Panelboard | Panelboard template | | NotSpools | eVolve_SpoolId is *Null* (no spool) | Exclude from spoolable deliverables |

These filters ensure only the intended elements populate each part template and deliverable.

Part Aliases (OPTIONAL, setup order: 7)

What it does: Maps alternate part names (e.g. vendor codes or old standards) to Stratus part names. This ensures consistency when users refer to parts by different names.

How to configure: Under *Admin* > *Company* > *Part Aliases*, add alias entries as needed. For example, if Revit family *eE_CF_Coupling* uses type names “EMT Compression” and “EMT Set Screw”, you might alias “EMT Compression Coupling” to “Coupling, EMT”.

Example table:
| Alias (from user data) | Canonical Part Name | |---|---| | “PVC Conduit (1″)” | **Rigid Conduit 1″** | | “Strut 1-5/8” | **Strut Channel 1-5/8″** | | “Panel 480V/3P” | **Panelboard 480V 3PH** |

These aliases help the Stratus BOM list use standardized names.

Import Mappings (REQUIRED, setup order: 8)

What it does: Defines how fields in incoming data (e.g. CSV exports or model syncs) map to Stratus fields. This is crucial for automated data import of parts, assemblies, and packages from Revit/eVolve.

How to configure: In *Admin* > *Project* > *Import Mappings* (or *Admin* > *Company* > *Import Mappings* if global), set up mappings between source columns/parameters and Stratus fields. You may have different mappings for “Parts”, “Assemblies”, and “Packages.” For example: map the Revit parameter *Length* to Stratus *Cut Length*, *Material* to *Material*, and *eVolve_CustomId* to *Assembly Number*.

Example mapping:

Stratus Field	Revit/eVolve Parameter or Export Column	Notes
Part Number	Combination of *Family Name* + *Type Mark*	e.g. `Family-Type`
Cut Length	*Length*	from Sols table
Diameter (Nominal)	*Diameter* (Revit) or *eV_ConduitSize* (shared)	mm or inches
Material	*Material* (instance or type)	text field
Spool ID	*eVolve_PrefabSpoolId* or *Custom Spool Tag*	spool grouping (see note)

Stratus Field	Revit/eVolve Parameter or Export Column	Notes
Level	*eVolve_LevelId* (shared parameter) [7]	building level
Custom ID	*eVolve_CustomId* (shared parameter) [7]	e.g. assembly ID

Note: For spools/packages, include fields such as **Spool Name** or **Package Name** mapped from the eVolve shared parameters (*eV_PackageId*, *eV_PrefabSpoolId*, or a tag). Stratus will use these to group parts into assemblies/prefabs.

Numbering Rules (REQUIRED, setup order: 9)

What it does: Controls numbering of deliverables (spools, assemblies, packages). Defines counters, prefixes, and reset rules. This is a subset of Naming & Numbering but applied at project scope.

How to configure: Under *Admin* > *Company* > *Naming and Numbering*, configure specific rules for each deliverable type:
- **Spool Numbering:** Define the format (e.g. `<ProjectCode>-S<000>`). Set the starting number and whether it resets per package or project.
- **Assembly Numbering:** If assemblies (e.g. panel assemblies) have their own IDs, define their prefix and increment.
- **Package Numbering:** Often the same as above spool; ensure **eV_PackageId** (or the chosen field) feeds Stratus package ID.

Example: If project "X123" uses 3-digit spools, Spool ID = `X123-001`, `X123-002`, … and Package ID = `X123-P001`, `X123-P002`. (These should match any naming templates set earlier.)

Deliverable Type Mapping (REQUIRED, setup order: 10)

What it does: Links Stratus deliverable categories (Parts, Cut List, Assemblies, Prefab Packages) to Stratus deliverable types. Essentially tells Stratus which templates to generate.

How to configure: Under *Admin* > *Company* > *Deliverable Type Mapping*, assign the "Parts" deliverable to the Parts/bom template, "Cut List" to the Cut List template, "Assembly" to the Assembly template, etc. (Exact navigation may vary by Stratus version.)

Example mapping:
| Stratus Deliverable Category | Stratus Template/Definition | |---|---| | BOM (Parts) | *Default BOM Report* | | Cut List | *Default Cutlist Report* | | Assembly | *Assembly Report* | | Prefab Package | *Prefab Package Report* |

This ensures when a Parts or Prefab package is exported, Stratus uses the correct output format.

Deliverable Mapping (REQUIRED, setup order: 11)

What it does: Specifies how parts are grouped into deliverables. For example, how parts are bundled into a "Bill of Materials" deliverable vs. an "Assembly". Often involves filter criteria or tags (e.g. all parts with the same *eV_CustomId* → one assembly).

How to configure: Under *Admin* > *Company* > *Deliverable Mapping*, create mappings or rules. For example, map the Revit parameter **eV_CustomId** (assembly identifier) to the Stratus Assembly Number field, so that all parts with the same CustomId go into the same assembly. Similarly, map *Package Name* (eV_PackageId) to Stratus Package field so spooled parts form a package.

Example:

Stratus Deliverable	Grouping Parameter	Usage
Assemblies	*eVolve_CustomId*	All parts with same CustomId → one assembly.
Prefab Packages	*eV_PackageId*	All parts with same PackageId → one prefab.
Cut Lists	(per Spool or Prefix)	Typically one cut list per spool.

This ensures Stratus correctly interprets the user's spool/package IDs (e.g. "PKG1", "PKG2" from the eVolve_PackageId parameter [8]) when building deliverables.

Deliverable Assembly (REQUIRED, setup order: 12)

What it does: Defines the assembly deliverable itself. An "Assembly" in Stratus is a set of parts (e.g. all electrical components in one panel or rack) compiled into one deliverable bundle.

How to configure: In *Admin* > *Company* > *Deliverable Assembly* (or under Project, depending on UI), set the default title, include drawings/schedules, and define assembly-specific options. For example, enable a "Assembly Tag" field to use *eV_CustomId* from eVolve as the assembly number.

Example: For panelboard assemblies, use the EVOLVE "Assembly" tag, which pulls *eV_CustomId*, *eV_AreaTypeId*, etc. into the assembly deliverable [6]. In Stratus Assembly settings, ensure the field "Assembly Name" maps to *Custom ID* (from Fields). Then each panel's parts with that Custom ID will form one assembly deliverable.

Deliverable Pieces (Prefab Packages) (REQUIRED, setup order: 13)

What it does: Defines the prefab/prefabrication package deliverable. A "Package" includes multiple spools or assemblies (e.g. a complete electrical rack or building section).

How to configure: Under *Admin* > *Company* > *Deliverable Pieces* (or *Prefab Packages*), set up package naming, default drawing templates, and which fields to include (like *Package ID*, *Category*, *Description*). Ensure Stratus has a field for **Package Name/ID** that receives the eVolve package parameter.

Example: All EVOLVE spools tagged with *eV_PackageId* become one Stratus package. Since *eV_PackageId* is a shared parameter used for packages [8], map Stratus *Package Number* to that field. Define any additional package fields (e.g. *Package Category* from Revit) and fill with values like "Electrical Panel".

Summary of Field Mappings

The tables below summarize example mappings between the user's Revit/eVolve content and Stratus fields for each deliverable type:

- **BOM (Parts)**: Key part parameters → Stratus BOM fields.
- **Cut List**: Part size/dimension fields → Stratus cut list fields.
- **Assembly**: Assembly grouping (eVolve custom IDs) → Stratus assembly number.
- **Package (Prefab)**: Package grouping (eV_PackageId) → Stratus package number.

Stratus Field	Example Revit/eVolve Source	Notes
Part Name	*Type Name* (Revit)	e.g. "EMT Coupling, 1"" (combined Family+Type)
Quantity	*Count* of identical parts	Auto-calculated in Stratus
Nominal Size/Width	*Diameter* or *Type Width*	From conduit/panel family parameters
Cut Length	*Length*	Part length for cut list
Material	*Type Material* (Revit)	e.g. "PVC", "Galv Steel"
Description	*Type Comments* or custom tag	Free text description field
Level	*eVolve_LevelId* (shared param) [7]	e.g. "Level 1"
Custom ID (Assembly)	*eVolve_CustomId* [7]	Used for grouping into assemblies
Spool Number	*Revit Spool Name* or *Custom Spool Tag*	E.g. "S-1001" from a routed conduit spool tag
Package Number	*eV_PackageId* (shared param) [8]	E.g. "PKG-01" from EVOLVE package creation
Package Category	*eP_PackageCategory* (if used)	Category/type of package (e.g. "Main Rack")

(These examples should be adjusted to match the actual parameter names and values in your Revit models. EVOLVE families provide shared params such as eV_PackageId and eV_CustomId that can populate these fields [6] [8].)

By carefully setting up each node above (Company Settings, Part Templates, Import Mappings, etc.) in this order, the user's EVOLVE/Revit content will flow into Stratus with all needed parameters aligned. Custom fields (like *Custom ID* and *Package ID*) ensure that stratus deliverables (BOMs, cut lists, assemblies, prefab

packages) are built using the company's existing spool and package identifiers, leveraging EVOLVE's shared parameters [6] [8].

Sources: Configuration details are guided by Stratus Admin UI and EVOLVE documentation. For example, EVOLVE families and parameters (like *eE_CF_Coupling* for conduit fittings [5] or *eVolve_CustomId* tags [6]) inform the tables above. The Stratus part templates UI (which shows *Properties, Cut List Mappings, BOM Mappings* tabs) is described in the Stratus release notes [4], and integration (App Keys) is noted in company admin instructions [1]. All field and filter mappings should be tested against the actual project data to ensure accuracy.

---

[1] [2] [3] [4] 01/25/2024 - v6.3.4 - STRATUS Knowledge Base - Confluence
https://gtpservices.atlassian.net/wiki/spaces/SK/pages/2213675009/01+25+2024+-+v6.3.4

[5] Current EVOLVE Families - EVOLVE Electrical Help
https://help-electrical.evolvemep.com/content/current-e-volve-families-list

[6] [7] eVolve Tag Families - EVOLVE Electrical Help
https://help-electrical.evolvemep.com/annotations-category/e-volve-tag-families

[8] Define Package - EVOLVE Electrical Help
https://help-electrical.evolvemep.com/evolve-electrical-spooling-prefabrication/create-prefab-package