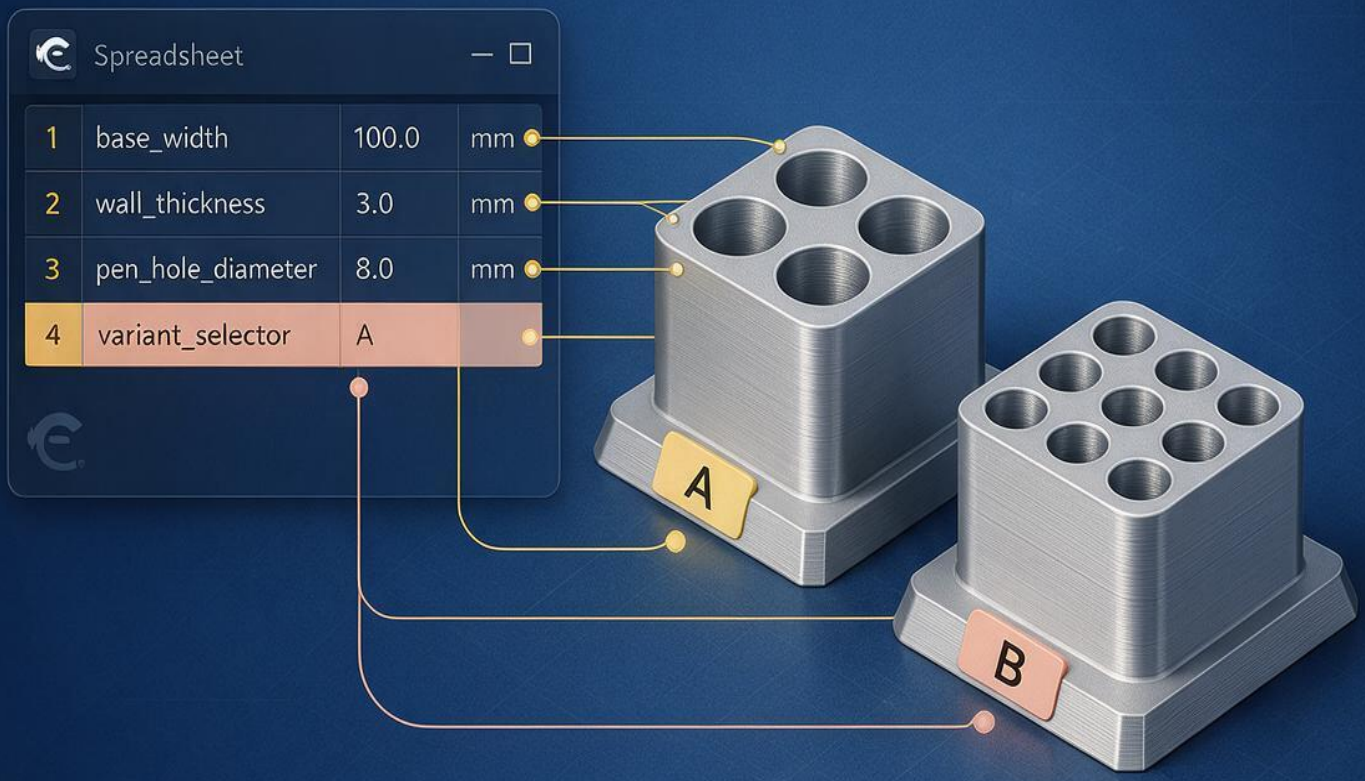
FreeCAD Parametric Design Guide

With focus on Design for Additive Manufacturing



Henk van Ekelenburg

1.1.2-01

FreeCAD Parametric Design Guide

With focus on

Design for Additive Manufacturing

Version 1.1.2-01

FreeCAD baseline: 1.1.2

Guide revision: 01

November 2026

Prepared by

Henk van Ekelenburg (Magimo)

© 2026 Magimo

Copyright and terms of use

© 2026 Magimo. All rights reserved.

This guide is made available free of charge. It may be used in normal engineering practice, including for commercial purposes, provided that this copyright notice and the other notices in this document are retained.

The guide may not be sold, licensed or otherwise made available for payment, either in its entirety or in part. This prohibition includes selling the document itself, bundling it with paid products or services and charging for access to it.

The guide may be forwarded or shared with others only in its entirety and unmodified. You may quote short sections, with credit and a link to the official source, in forums, wikis and teaching material. Partial reproduction, extraction, adaptation or redistribution of individual sections is not permitted without prior written permission from the author.

Readers are encouraged to obtain the guide directly from the author's official distribution channel, rather than relying on forwarded copies, so that they receive the most recent version and revision notices.

Revision notice

This guide is versioned against the FreeCAD release on which its guidance, screenshots, expressions and workflows have been verified. The version number therefore identifies both the applicable FreeCAD baseline and the revision of this guide for that baselineFor example, a guide based on a hypothetical FreeCAD 30.2 would be numbered 30.2-01 for its first revision and 30.2-02 for the next revision on the same baseline. A new FreeCAD baseline results in a new guide version, such as 30.3-01 in this hypothetical example.

This work is subject to regular revision, correction and expansion. If you received it from a peer, colleague or friend, your copy may not be current. Information and process recommendations in this rapidly developing engineering discipline can become outdated; always check the latest version and validate guidance before applying it to a functional or safety-relevant part.

Readers are encouraged to report errors, suggest additional subjects, identify outdated or unclear material and submit useful references or test results for consideration in future revisions. Practical feedback, test results, references and well-documented examples are especially welcome. Submissions may be considered for inclusion in a future revision, but their acceptance, editing and attribution remain at the author's discretion.

Reader feedback and contributions: magimo.freecad@magimo.eu

Readers are encouraged to submit suggestions for additional subjects, identify errors, report unclear or potentially outdated information and propose sections that require correction or adjustment. Practical feedback, test results, references and well-documented examples are especially welcome. Submissions may be considered for inclusion in a future revision, but their acceptance, editing and attribution remain at the author's discretion.

Disclaimer

This guide is provided for educational and engineering guidance purposes only. It does not replace application-specific engineering analysis, material and machine qualification, testing, risk assessment, regulatory compliance or professional advice.

Additive-manufactured parts can contain process-dependent defects and may behave differently from bulk-material data. The author and publisher accept no responsibility for loss, damage, injury or other consequences arising from reliance on this guide or from the manufacture or use of components designed with it.

The creator, manufacturer and user remain responsible for selecting suitable materials and processes, validating the finished part and determining whether it is appropriate for its intended application.

ACKNOWLEDGEMENTS **12**

1 PURPOSE AND ENGINEERING POSITION **13**

1.1 The problem: models leave their creator 13
1.2 Scope and intended reader 15
1.3 What this guide covers – and does not cover 16
1.3.1 What this guide covers 16
1.3.2 What this guide does not cover 17
1.4 The parametric model as a technical contract 18
1.5 Controlled freedom 18
1.6 How to use this guide 19
1.6.1 General flow 19
1.6.2 Map 20
1.6.3 Formats used 20
1.7 FreeCAD version used throughout this guide 21

2 DEFINE THE DESIGN BOUNDARY **22**

2.1 Design intent before dimensions 22
2.2 Define the creator–user boundary 24
2.3 Define the supported configuration domain 26
2.4 Classify parameters by role 28
2.4.1 Context parameters 28
2.4.2 Choice parameters 29
2.4.3 Engineering parameters 29
2.4.4 Derived parameters 30
2.4.5 Driving and dependent values 30
2.5 Expose decisions, not implementation 31
2.6 Design for explicit assumptions 32
2.6.1 Counter the curse of knowledge 32
2.6.2 What to assume about the user 33
2.6.3 User interface 34
2.6.4 Manufacturing and use information 34
2.6.5 Practical test 34

3 BUILD THE PARAMETER INTERFACE **35**

3.1 The Spreadsheet as a control layer 35
3.1.1 The user configuration layer 35
3.1.2 The creator control layer 36
3.1.3 Separation from geometry 36
3.1.4 Possible implementations 37
3.1.5 Alternative parameter containers in FreeCAD 38
3.2 One source of truth 39

3.3 Names, aliases, units and meaning 40
3.3.1 Names and aliases 41
3.3.2 Units 41
3.3.3 Descriptions and documentation 42
3.4 Group parameters by role and visibility 43
3.4.1 Arrange by information flow 43
3.4.2 Separate user-facing and protected areas 43
3.4.3 Apply consistent editability 44
3.4.4 Review each parameter 44
3.5 Defaults, limits and choices 45
3.5.1 Defaults are part of the design 45
3.5.2 Define limits explicitly 45
3.5.3 Use discrete choices where they fit 46
3.5.4 Keep continuous inputs when reality is continuous 47
3.5.5 Do not use limits to conceal uncertainty 47
3.5.6 Record the parameter definition 47
3.5.7 Before exposing a parameter 48
3.6 Progressive disclosure and advanced controls 48
3.6.1 Basic and advanced layers 49
3.6.2 Keep complexity out of the basic path 49
3.7 Review the parameter interface 50
3.7.1 Interface review checklist 50
3.7.2 Do not connect geometry prematurely 51

4 BUILD A ROBUST FREECAD MODEL 52

4.1 Robustness means reliable behaviour under change 52
4.1.1 A nominal model is not a tested model 52
4.1.2 Geometry alone is not enough 52
4.1.3 Robustness has a defined domain 53
4.1.4 Design for controlled failure 53
4.1.5 Robustness is an engineering property 53
4.2 Stable references and the Topological Naming Problem 54
4.2.1 The failure mechanism 54
4.2.2 Prefer deliberate references 55
4.2.3 Separate architectural and local references 55
4.2.4 Keep references semantically clear 55
4.2.5 Test reference stability 55
4.3 Skeletons, master sketches and functional reference geometry 56
4.3.1 Master sketches define the main relationships 56
4.3.2 Use a small number of purposeful references 56
4.3.3 Establish a deliberate dependency direction 57
4.3.4 Reference function, not only shape 57
4.3.5 Connect downstream geometry deliberately 57

© 2026 Magimo

4.3.6 Do not confuse a skeleton with the finished part 57
4.3.7 The architectural rule 57
4.4 Constraint strategy 58
4.4.1 Fully constrained, but not over-constrained 58
4.4.2 Driving and driven dimensions 59
4.4.3 Anchor geometry to functional references 59
4.4.4 Keep each sketch comprehensible 59
4.4.5 Avoid constraints that conceal the design intent 59
4.4.6 Test the sketch as a system 59
4.5 Expressions: centralise logic, simplify geometry 60
4.5.1 One relationship, one location 60
4.5.2 Keep sketch expressions short 60
4.5.3 Keep geometric relationships in the sketch 60
4.5.4 Avoid raw cell references and duplicated constants 60
4.5.5 Avoid circular and opaque dependency chains 61
4.5.6 Expressions should support testing 61
4.5.7 Practical rule 62
4.6 Feature-tree and Body strategy 62
4.6.1 Optional features and configuration-dependent geometry 63
4.6.2 Alternative topology Bodies 64
4.6.3 Visibility Toggle as a configuration selector 66
4.7 Scaling versus genuine parametric behaviour 66
4.8 Performance and maintainability 67

5 ENCODE MANUFACTURING IN THE MODEL 68

5.1 DfAM is a design input, not a slicer correction 68
5.2 Process context as parameter input 68
5.3 Material and process variation 69
5.3.1 Shrinkage and dimensional change 69
5.3.2 Moisture sensitivity 69
5.3.3 Layer bonding and mechanical behaviour 70
5.3.4 Stiffness, creep and temperature 70
5.3.5 Material profiles and configuration control 70
5.4 Orientation, anisotropy and load path 71
5.4.1 Orientation and anisotropic strength 71
5.4.2 Surface finish and dimensional behaviour 72
5.4.3 Supports and orientation-dependent geometry 72
5.4.4 Orientation as part of the release contract 73
5.5 Walls, features, bridges and support strategy 73
5.5.1 Walls and shells 73
5.5.2 Minimum features 74
5.5.3 Bridges and overhangs 74
5.5.4 Support strategy 74

5.5.5 Rule hierarchy 75
5.6 Fits, clearances, interfaces and tolerance stack-up 75
5.6.1 Define the interface first 76
5.6.2 Make clearances process-aware 76
5.6.3 Keep interfaces consistent 77
5.6.4 Tolerance stack-up 77
5.6.5 Interfaces beyond holes and pins 77
5.6.6 Scaling and tolerance behaviour 78
5.6.7 Release requirement 78
5.7 Hard rules, soft guidance and exceptions 78
5.7.1 Hard rules 78
5.7.2 Soft guidance 79
5.7.3 Human review 79
5.7.4 Exceptions must be explicit 80
5.7.5 Rule precedence 80
5.8 Manufacturing-aware profiles 80
5.8.1 What a profile contains 80
5.8.2 Keep the base model intact 81
5.8.3 Profiles contain coupled assumptions 81
5.8.4 Profile status 82
5.8.5 Profile selection and output 82
5.8.6 Embedded and external profiles 82
5.8.7 Do not overstate profile coverage 83

6 VERIFY, CONTROL AND RELEASE 84

6.1 What “validated” means for a parametric product 84
6.1.1 Status terms 84
6.1.2 Validated 85
6.1.3 Supported 86
6.1.4 Released 86
6.1.5 Other status terms 87
6.1.6 Relationship between statuses 87
6.1.7 Status and configuration selection 88
6.1.8 Status and evidence records 88
6.2 Boundary testing and representative test cases 88
6.2.1 Test the parameter boundaries 88
6.2.2 Test coupled extremes 89
6.2.3 Test representative configurations 89
6.2.4 Check more than recompute status 89
6.3 Print trials and functional evidence 90
6.3.1 Record the manufacturing conditions 90
6.3.2 Record observations and failures 90
6.3.3 Measure critical dimensions 91

© 2026 Magimo

6.3.4 Test function under defined conditions 91
6.3.5 Connect evidence to the supported domain 91
6.3.6 Tolerance-ladder coupon 92
6.4 Error handling and controlled failure 95
6.4.1 Four controlled responses 95
6.4.2 Detect errors at the parameter layer 95
6.4.3 Do not hide partial failure 95
6.4.4 Make messages actionable 96
6.4.5 Controlled failure and human review 96
6.5 Configuration management and change control 96
6.5.1 Identify the configuration elements 97
6.5.2 Version the model according to behaviour 97
6.5.3 Preserve parameter compatibility 97
6.5.4 Manage profiles separately 97
6.5.5 Use a compatibility matrix 98
6.5.6 Perform change-impact analysis 98
6.5.7 Release immutable combinations 98
6.6 In-model documentation 98
6.6.1 Document during design 98
6.6.2 Document parameters where they live 98
6.6.3 Document protected logic 99
6.6.4 Document the model itself 99
6.6.5 Document status and limitations 99
6.6.6 Keep documentation maintainable 100
6.7 Packaging the released file 100
6.7.1 Start from a release copy 100
6.7.2 Clean the model tree 100
6.7.3 Remove or isolate debugging material 100
6.7.4 Control dependencies 101
6.7.5 Include supporting files 101
6.7.6 Identify versions and status 101
6.7.7 Perform the release check 102

7 HANDOVER, PLATFORMS AND DESIGN-SYSTEM THINKING 103

7.1 The Technical File 103
7.1.1 What the Technical File records 103
7.1.2 Internal record versus released documentation 105
7.2 The manufacturing handover 106
7.2.1 Material and printer 107
7.2.2 Orientation and load direction 107
7.2.3 Walls, shells and infill 107
7.2.4 Supports, cavities and seams 107
7.2.5 Fits, tolerances and scaling 108

© 2026 Magimo

7.2.6 Compact manufacturing manifest 108
7.3 The use and safety handover 109
7.3.1 Structural limits 109
7.3.2 Environmental and chemical limits 109
7.3.3 Assembly and installation 110
7.3.4 Foreseeable misuse 110
7.3.5 Inspection and retirement 110
7.4 External parameter interfaces and automation 111
7.4.1 Preserve the parameter roles 111
7.4.2 Define an interface contract 111
7.4.3 Validate before generation 111
7.4.4 Handle formats carefully 112
7.4.5 Keep generation deterministic 112
7.4.6 Separate inputs from outputs 112
7.4.7 Preserve the design intent during automation 112
7.4.8 External parameter libraries 113
7.5 Platform-neutral distribution requirements 115
7.5.1 Minimum release package 115
7.5.2 The parameter schema is part of the product 115
7.5.3 Minimum operational behaviour 116
7.5.4 Separate creator and user packages 116
7.6 A distribution platform implementation 116
7.6.1 Generic concept to platform mapping 116
7.6.2 Use stable identifiers 117
7.6.3 Keep validation in the model boundary 118
7.6.4 Preserve status and evidence 118
7.6.5 Platform-specific material belongs in an adapter layer 118
7.7 From shape to controlled design system 119

APPENDIX A MINIMUM CHECKLIST FOR SIMPLE SHARED PARTS 120

A.1 Purpose 120
A.2 Checklist 120
A.3 Where to put it 121
A.4 When to move to the full process 121

APPENDIX B PRACTICAL FREECAD IMPLEMENTATION OF THE SPREADSHEET CONTROL LAYER 122

B.1 Purpose and scope 122
B.2 Minimum sheet structure 122
B.3 Common Spreadsheet conventions 123
B.3.1 General 123
B.3.2 Colour scheme 124
B.4 UIParams: the user configuration sheet 125

B.5 CMParams: the creator-control sheet 126
B.6 MaterialProfile and ProcessAspects 127
B.7 UseConstraints and Validation 128
B.8 ReleaseInfo and DocsIndex 128
B.9 Practical implementation techniques 129
B.10 Final implementation check 130

APPENDIX C CONFIGURATION TABLES IN FREECAD 131

C.1 Purpose 131
C.2 When to use one 131
C.3 What it does not replace 132
C.4 Basic concepts 133
C.4.1 Configuration Key 133
C.4.2 Selector 133
C.4.3 Result values 134
C.5 Table orientation 134
C.5.1 Horizontal table 135
C.5.2 Vertical table 135
C.6 Table layout and aliases 135
C.7 Retrieving values in FreeCAD 136
C.7.1 Native FreeCAD Configuration Tables 137
C.7.2 Direct alias references 138
C.7.3 Centralised conditional expressions 138
C.7.4 Scripted or external lookup 139
C.7.5 Independent selectors and dependent choices 139
C.7.6 Selection and lookup checks 139
C.8 Recommended architecture 140
C.9 Worked example: parametric pen holder 141
C.9.1 User inputs 141
C.9.2 Configuration Key 141
C.9.3 Geometry rules 142
C.9.4 Calculated values 142
C.9.5 Counts and key 143
C.9.6 Hole diameters 143
C.9.7 Hole positions 143
C.9.8 Product envelope 143
C.9.9 Validation status 144
C.9.10 Worked example: configuration 111 144
C.9.11 Configuration table 145

APPENDIX D BIT HOLDER: DESIGN BASIS AND CALCULATIONS 147

D.1 Purpose 147
D.2 Design intent and requirements 147

© 2026 Magimo

D.3 Assumptions and design boundaries 147
D.4 Design rationale and calculations 148
D.4.1 Layout 148
D.4.2 Loads and moments 148
D.4.3 Hook on the cleat 149
D.4.4 What the numbers mean 149
D.5 Validation evidence 150
D.6 Limitations and open issues 150
D.7 Revision history 151

APPENDIX E ADVANCED FREECAD TOOLBOX 152

E.1 Purpose 152
E.2 Spreadsheet architecture that scales 153
E.2.1 When to split the control layer 153
E.2.2 Process parameters for FDM geometry rules 155
E.3 Expression patterns that matter in any domain 156
E.3.1 Guardrail functions: `min()`, `max()` and `abs()` 157
E.3.2 Snapping to process steps: `floor()`, `ceil()` and `round()` 158
E.3.3 Snap wall thickness to process multiples 159
E.3.4 Checking alignment with mod() 160
E.3.5 Tolerance-stack checks 161
E.3.6 Directional compensation and anisotropic scaling 167
E.3.7 Geometric checks from driving dimensions 168
E.3.8 Conditional logic and Boolean rules 170
E.3.9 Unit stripping and unit attachment 177
E.3.10 Named ratios and intermediate values 177
E.3.11 Integer counts and leftover span 177
E.3.12 Lists and indexed selection 178

APPENDIX F PARAMETRIC CONTROLS OUTSIDE THE SPREADSHEET 182

F.1 Purpose 182
F.2 Switching a feature on or off: Suppressed 182
F.2.1 What it does 182
F.2.2 When to use it 182
F.2.3 How to implement it 182
F.2.4 Watch-outs 182
F.3 A drop-down in a VarSet 183
F.3.1 What it does 183
F.3.2 When to use it 183
F.3.3 How to implement it (example) 183
F.3.4 Watch-outs 183

© 2026 Magimo

Acknowledgements

This guide was not written alone. Many FreeCAD users, engineers and makers read drafts, tested examples and sent comments. Their questions showed where the text was unclear, and their suggestions and corrections made it more accurate. I am grateful to all of them.

Special thanks go to Vamsee Achanta and Leonardo Gon. They helped shape the overall concept of the guide and proofread Chapters 1–7. Their comments on structure and consistency made the text clearer and more coherent.

Finally, thanks to the FreeCAD developers and community. Without their work there would be no FreeCAD to write about.

Any remaining errors are my own. Comments and corrections are welcome.

Henk van Ekelenburg

November 2026

www.magimo.eu

magimo.freecad@magimo.eu

1 Purpose and Engineering Position

This chapter explains why a parametric model that leaves its creator needs more than adjustable dimensions. It sets out who the guide is for, what it covers and how to use it, and it defines the terms the later chapters rely on: creator, user, technical contract and supported domain.

1.1 The problem: models leave their creator

Drawings and sketches provided a way to record and communicate design knowledge. They allowed a design to be examined, improved and reproduced by someone who had not created it. When a drawing included dimensions, materials, tolerances, fits, assembly details and manufacturing notes, it became a means of transferring design intent.

CAD extended this process. A CAD model can describe geometry in greater detail than a drawing. Constraints and relationships can also preserve part of the reasoning behind the geometry. Parametric design goes one step further by describing a controlled set of possible shapes and the relationships that govern them. This creates both an opportunity and a risk. Parametric design can improve efficiency, reuse and adaptation, but only when the parameters represent meaningful design decisions and the relationships remain valid when values change.

Additive manufacturing introduces another requirement. The model must reflect how the part will be produced: by depositing material in layers with a particular bead width, layer height, thermal history, orientation, degree of anisotropy and support strategy. These conditions affect geometry, dimensional accuracy, strength, surface quality and service life.

© 2026 Magimo

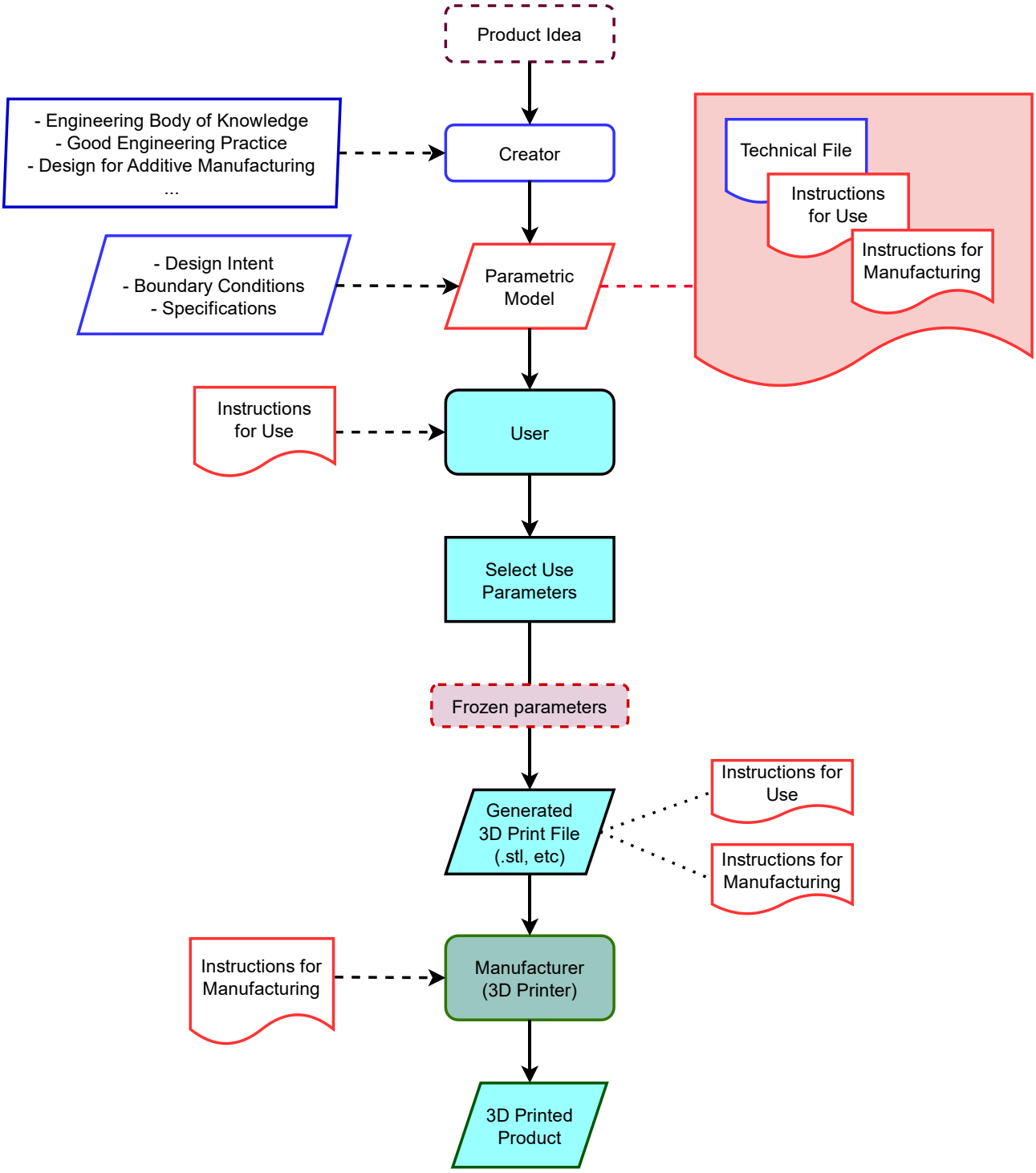


Figure 1–1: Development, validation, documentation and release flow for a distributed parametric design

If these conditions are not understood and encoded in the design, parametrisation does not remove uncertainty. The model does not blush or object. It will confidently generate geometry that is mathematically valid but physically senseless, unmanufacturable or outside the validated design domain. The more poorly defined parameters it exposes, the easier it becomes to create configurations that are possible in CAD but unfit for their intended use.

The same problem appears when the model leaves its creator. The creator may know which parameters are safe, which combinations are untested, which orientation is required and which dimensions are derived rather than independent. A new user sees only the file, the interface and

the available choices. If the design intent and manufacturing assumptions have not been made explicit, the user must reconstruct them through trial and error.

A distributed parametric model therefore needs more than adjustable dimensions. Its creator must:

- understand the additive-manufacturing process and its limits;
- encode relevant manufacturing knowledge in the geometry, parameters and expressions;
- distinguish supported configurations from merely possible ones; and
- communicate design intent, manufacturing requirements and limitations.

This guide addresses that problem. It treats parametric Design for Additive Manufacturing (DfAM) not as a set of CAD tricks or slicer settings, but as passing design intent, in a controlled way, from the creator to the user, the machine and the manufacturing setup.

1.2 Scope and intended reader

This guide is for FreeCAD users who want other people to configure, manufacture and use their parametric models with reasonable confidence. It is aimed primarily at engineers and technically experienced creators who create functional parts using extrusion-based polymer additive manufacturing, such as FDM or FFF.

The guide assumes practical experience with FreeCAD. The reader should be able to create and edit sketches, apply geometric and dimensional constraints, use expressions and aliases, work with Bodies and features, and organise parameter values in the Spreadsheet workbench. The guide does not explain basic operations such as creating a sketch, applying a pad or opening a Spreadsheet.

The focus is what happens after these basic operations are understood. The guide examines how to structure the model, decide which parameters should be exposed, encode design and manufacturing constraints, and test the resulting behaviour across a defined range of configurations. It is concerned with the difference between a model that works for its creator and a parametric product that remains understandable and dependable when operated by someone else.

The principles in this guide are platform-neutral. They apply whether a model is shared as a FreeCAD file, generated by a script, managed in a company library, published through a print service or distributed through a web-based platform (you do not need a platform to use this guide). In every case, the central question is:

> Can another competent person change the intended inputs and obtain a result that remains within the known design and supported domain? Yes, but only when the model's inputs, rules, limits, failure behaviour, evidence and handover are explicit. Otherwise the result depends on knowledge that only the creator has.

The method is not limited to printing. The parameter roles, the contract and the support levels apply equally to machined, laser-cut and sheet-metal parts; only the examples are print-specific.

© 2026 Magimo

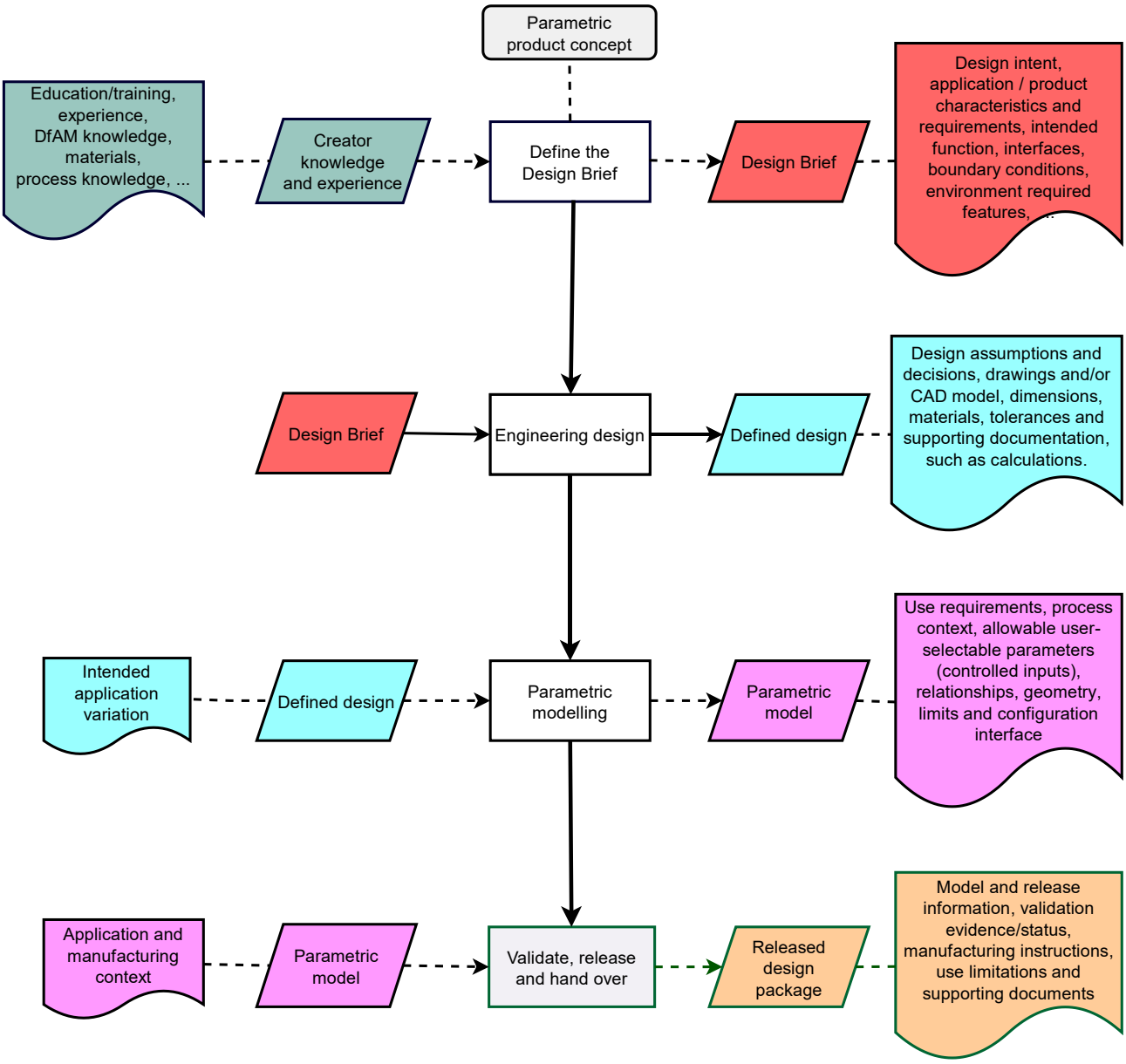


Figure 1–2: Input–process–output chain from creator knowledge to a released parametric DfAM product. Solid arrows show the main process sequence; dashed connectors associate each process with its input information and output artefacts. The documents shown are representative examples.

1.3 What this guide covers – and does not cover

1.3.1 What this guide covers

This guide covers the design of parametric FreeCAD models intended for repeated configuration, manufacture and use. It focuses on the relationship between design intent, parameter architecture, model structure, additive-manufacturing constraints and the behaviour of the finished part. The main subjects are:

- separating context, choice, engineering and derived parameters (§2.4);
- structuring the Spreadsheet and expressions as a control layer;
- building robust sketches, references, Bodies, features and model trees;

- preserving function when dimensions and configurations change;
- encoding relevant FDM/FFF constraints in the design;
- managing clearances, tolerances, orientation, anisotropy, supports and material assumptions;
- validating configurations through recompute checks, print trials, dimensional measurements and functional tests; and
- documenting, packaging and handing over the model so that its design intent remains available after it leaves its creator.

The model is shared as a FreeCAD file (.FCStd): a ZIP archive holding the model as XML plus shape files. The user needs FreeCAD to change it; other CAD programs can import only an exported shape (STEP, STL) without its parameters and history. Chapter 7 covers the release package and neutral parameter formats such as JSON and CSV.

Users do not change the model during its life. Feedback from use goes back to the creator, who changes and re-releases the model under change control (see §6.5).

1.3.2 What this guide does not cover

This guide is not a beginner's FreeCAD tutorial. It does not explain every workbench or provide click-by-click instructions for creating basic sketches, pads, pockets or Spreadsheets. It assumes that the reader can perform these operations and wants to understand how to combine them into a dependable parametric system.

Nor is it a slicer manual, a catalogue of universal print settings or a replacement for printer- and material-specific validation. FDM/FFF behaviour depends on the machine, nozzle, process settings, material, environment and part geometry. Numerical values and design rules given in this guide should therefore be treated as engineering starting points or examples unless they are supported by appropriate testing.

The guide also does not claim that a model is suitable for manufacture or use merely because it recomputes without an error. A successful recompute shows only that the CAD system has generated a result. It does not prove that the result is printable, dimensionally suitable, structurally adequate, durable in service or appropriate for a particular application. Each of those claims needs more evidence than the one before.

The guide does not prescribe one modelling style for every FreeCAD project. The practices described in this guide should be applied proportionately. A simple bracket designed, printed and used by one person in a domestic setting may not require a separate Technical File, an external parameter interface, a formal release package or extensive configuration testing. For such a part, a short record of the intended use, material, orientation, important dimensions and known limitations may be sufficient.

The full design-system approach becomes more valuable when a model is configurable, reused across different applications, manufactured by other people, distributed beyond its creator or used where failure could cause significant damage or injury. The purpose of this guide is therefore not to impose the complete process on every part. It is to help the creator select a level of structure, validation and documentation proportionate to the consequences of failure, the number of users, the variability of the configuration and the distance between the creator and the manufactured result.

> **Practical rule**: use the simplest process that provides sufficient confidence for the intended use, while increasing control and evidence as configuration complexity, distribution and consequences of failure increase. For a simple shared part, the minimum checklist in Appendix A is enough.

© 2026 Magimo

The guide does not replace engineering judgement; it helps make that judgement explicit, reusable and traceable through the parameters, relationships, geometry, validation records and handover information.

> The objective is not to make every model capable of every variation. It is to make the intended variations understandable, controlled and justified.

1.4 The parametric model as a technical contract

A parametric model is more than a collection of dimensions that can be changed. It is an agreement between the person who creates the model and the person who operates it. Where the boundary between creator and user lies, is set out in §2.2.

One person can hold several roles; the responsibilities stay separate. The model translates these inputs into geometry intended to remain within the declared design limits. This agreement is not a legal contract. It is a technical contract: a clearly defined relationship between inputs, rules, outputs, assumptions and limitations. It exists whether the model is used privately, shared with a colleague, published in a library or distributed through a platform. A sound technical contract answers at least five questions:

- What is the part intended to do?
- Which values may the user change?
- What relationships and constraints govern the result?
- Under which manufacturing and service conditions has the result been designed or validated?
- What happens when an input falls outside those conditions?

The contract must be reflected in the model itself. A parameter named `wall_thickness` is not sufficient if the user can reduce it below a manufacturable or structurally acceptable value. A material selector is not sufficient if changing the material does not update the relevant clearance, wall or process assumptions. A model that permits these changes without warning exposes options without defining their consequences.

An experienced user may be able to edit a Spreadsheet, change a sketch or override a value, but those actions do not automatically belong in the intended user interface (See §2.6.3). The relevant question is what the creator has defined, tested and agreed to support. A user may select a fit class, for example, without being expected to calculate the corresponding clearance. A user may select an approved material without editing the material properties or process compensation associated with it.

A parametric model is complete when its purpose, permitted inputs, governing relationships, assumptions and limits are explicit enough for another person to configure and use it as intended. The contract states what may change. The next section asks how much should change.

1.5 Controlled freedom

Parametric design is often presented as the ability to change almost anything. That may be technically attractive, but it is not necessarily good engineering. A model that exposes every dimension can transfer the burden of design decisions, validation and error detection to the user.

Every exposed parameter creates additional possible configurations and interactions. A change in length may affect wall thickness, clearance, feature spacing, load paths, print orientation or the space available for a fillet. If these relationships are not controlled, the number of possible outcomes can grow faster than the creator can reasonably test.

© 2026 Magimo

The objective is therefore not maximum flexibility. It is controlled freedom: allowing the user to change the aspects that matter while protecting the relationships that make the result functional and manufacturable. A controlled parametric model generally follows four principles:

- Expose values that describe the user's real situation, such as a mating diameter, required length, hand size, available space or selected material.
- Expose deliberate choices that produce meaningful and understandable differences, such as a fit class, edge style or approved pattern.
- Derive values that should follow from other inputs, such as wall thickness, transition length, hole clearance or feature spacing.
- Protect values that exist to maintain structural integrity, geometric stability or printability.

For example, a configurable grip may allow the user to enter hand width, select a fit class and choose between two approved edge styles. The model may then derive the grip diameter, wall thickness, internal clearance, fillet radius and feature spacing. Exposing all these values would not necessarily improve the design; it could instead transfer engineering decisions to the user that the creator is better placed to encode and validate. Therefore, expose a parameter only when it has a clear purpose for the user, a known range and a tested effect. The checklist in §3.5.7 tests this; the answers become part of the technical contract.

If these questions cannot be answered, the parameter should normally remain within the model's internal logic until its purpose, limits and behaviour are understood.

> The strongest parametric models are not those with the most adjustable values. They provide enough freedom to cover the intended use cases while keeping the result within a design and manufacturing domain that the creator understands and is prepared to support. The aim is not to expose every degree of freedom, but to expose the right ones and control the rest.

1.6 How to use this guide

1.6.1 General flow

This guide moves from general engineering principles to FreeCAD implementation, manufacturing-aware design, validation and distribution. The order is deliberate. A robust parametric model cannot be designed properly until its purpose, users, permitted variation and manufacturing context have been defined.

The guide follows the development and release path of a distributed parametric design. This path includes both the technical development of the model and the documents needed to define, validate, release and hand it over.

The flowchart is not intended to prescribe one software architecture. It shows the main decisions, information flows and documents that must be controlled as a design moves from creator knowledge to a configured and manufactured result. The later chapters examine these stages in more detail, including the model, the Technical File, validation evidence, release information and handover documentation.

For a new model, the most useful reading path is generally:

1. Define the design boundary and supported configuration domain.
2. Identify context inputs, user choices, engineering values and derived results.
3. Build the parameter interface and decide what should remain protected.

© 2026 Magimo

4. Structure the FreeCAD model so that its references, constraints, expressions and feature tree remain stable under change.
5. Encode relevant manufacturing conditions and limitations.
6. Test the model, record the evidence and prepare the handover information.

For an existing model, use the guide as a review framework. Start by asking what the model is intended to do, who will operate it and which configurations have actually been tested. Then review the parameter interface, model robustness, manufacturing assumptions, validation evidence and release documentation.

1.6.2 Map

The main chapters provide the principles and reasoning. Detailed tables, templates, implementation patterns and worked examples are in the appendices. Cross-references point to the relevant one. The guide can therefore be used as:

- a reference while designing a new parametric model;
- a review framework for improving an existing FreeCAD model;
- a release checklist before distributing a configurable design.

Table 1–1: Navigation guidance

If you are…	Start with	Then use
Preparing a simple part quickly	Appendix A	Chapter 2, if the part grows
Creating a new parametric product	Chapters 2–4	Appendix B, Appendix C and Appendix D
Preparing a model for manufacture	Chapter 5	§6.3.6 (tolerance-ladder coupon) and Appendix D
Distributing or handing over a model	Chapters 6–7	§6.7 (packaging) and Appendix B
Improving an existing model	Chapters 4 and 6	§3.7 (interface review) and Appendix A
Building reusable modules or variants	§4.6 and Chapter 7	§7.4.8 (external libraries) and Appendix C
Working with Configuration Tables	Appendix C	§C.9 (worked example)
Looking for a complete worked example	The running-example boxes, Chapters 2–7	Appendix D
Setting up the Spreadsheet control layer	Chapter 3	Appendix B (conventions), then Appendix E (expression patterns)
Using parametric controls outside the Spreadsheet	§3.1.5	Appendix F

1.6.3 Formats used

- Chapters and appendices are referred to by name (Chapter 4, Appendix A), sections by § (e.g. §2.4). In the electronic version, each reference is a link to that place.

- Aliases, properties and identifiers are shown in code format, without quotation marks, for example `wallThickness`, `LinkCopyOnChange` and `MOUNT_STD`.[1]
- Engineering terms are written in ordinary text, for example wall thickness, configuration key, material profile and active result.
- FreeCAD functions are written in lowercase with parentheses, for example list(), `floor()` and `round()`.
- FreeCAD object types are capitalised when referring specifically to the object type, for example Body, Spreadsheet and App Link.
- Statuses are capitalised when used as defined states, for example Draft, Validated, Supported and Released.

1.7 FreeCAD version used throughout this guide

The guide version follows the FreeCAD baseline on which it is issued. Version 1.1.2-01 is the first revision of this guide based on 1.1.2. Corrections or additions issued for the same FreeCAD baseline are identified by an incremented guide-revision number, such as 1.1.2-02. When the guide is updated for a later FreeCAD release, the version number changes accordingly, for example 26.3-01[2].

Features, expressions, user-interface details and workflow behaviour may differ in other FreeCAD versions. Where a capability is version-dependent, this is stated in the relevant section. Readers using a different FreeCAD version should verify that the described functions and behaviour are available before applying the guidance to a functional or safety-relevant design.

FreeCAD normally opens files made in earlier versions; a file saved in a newer version may not open in an older one. Since FreeCAD 1.0 (November 2024), mitigation of the Topological Naming Problem has made models more stable under edits, and from 26.3, FreeCAD releases three times a year on fixed dates. Before moving a released model to a new FreeCAD version, recompute its released configurations and repeat the checks outlined in §6.2.

1 The examples in this guide use snake_case (wall_thickness), camelCase (fitClearance) and PascalCase (LineWidth). All three are valid FreeCAD aliases, and readers will meet all three in shared models. Within one model, choose one style and use it consistently: mixed styles make expressions harder to read and aliases easy to mistype.
A useful refinement is to let the style carry meaning, for example snake_case for inputs and engineering values and PascalCase for check results and status values (EnvelopePass, ConfigStatus). The style then shows what kind of value an alias holds. If you do this, state the rule in the model's documentation and apply it consistently.

2 This example reflects the new versioning scheme; this edition uses the FreeCAD baseline and guide revision shown above.

© 2026 Magimo

2 Define the Design Boundary

This chapter sets the design boundary: what the creator keeps, what the user may decide and which configurations the creator will support. Write these down before you draw any geometry, because every later choice of parameters depends on them.

2.1 Design intent before dimensions

A parametric model should not begin with a list of dimensions. It should begin with a statement of what the part must do, under which conditions and which aspects of its behaviour are allowed to vary.

This distinction is easy to overlook because CAD software presents geometry as the immediate design task. The first visible activity is often the creation of a sketch, followed by the addition of dimensions and constraints. In a configurable model, however, these dimensions are consequences of earlier decisions. If those decisions are not defined first, the parameter list becomes an accidental collection of values rather than a representation of design intent. Before you draw the first sketch, write down:

- the function of the part;
- what it must not be used for;
- the interfaces with other parts, tools or users;
- the relevant loads and load directions;
- the environmental and service conditions;
- the intended manufacturing process and orientation;
- the dimensions that must remain fixed;
- the dimensions or features that may vary;
- the relationships that must remain true when they vary;
- the conditions under which the design is no longer valid; and
- what evidence supports the permitted configuration range.

For a configurable component, design intent also includes the intended range of variation. A model designed for three cable diameters has a different engineering basis from a model intended to accept any cable diameter entered by the user. The first can be designed and tested around defined interfaces. The second requires a broader set of rules, limits and validation evidence.

The intended variation must preserve the functional relationships that matter. Changing a length, grip size or interface dimension may require corresponding changes to wall thickness, feature spacing, clearances, load paths or orientation.

This does not need to be a lengthy specification for every project. A small internal part may need only a short design record. A distributed parametric product requires more because the creator's informal knowledge will no longer be available to correct misunderstandings. In practice, the rule is straightforward:

> Do not ask first: "Which dimensions should be parameters?" Ask: "Which aspects of the design are intended to vary and what must remain true when they do?"

© 2026 Magimo

Running example – a bit holder for a French cleat.

A French cleat is a popular system for flexible storage in workshops and garages. It uses two strips with a 45° bevel: one is fixed to the wall with the bevel facing up and towards the wall, and the stored item carries a matching strip that hooks over it. Gravity pulls the item down onto the bevel and locks it against the wall, yet it can be lifted off and rehung anywhere along the cleat.

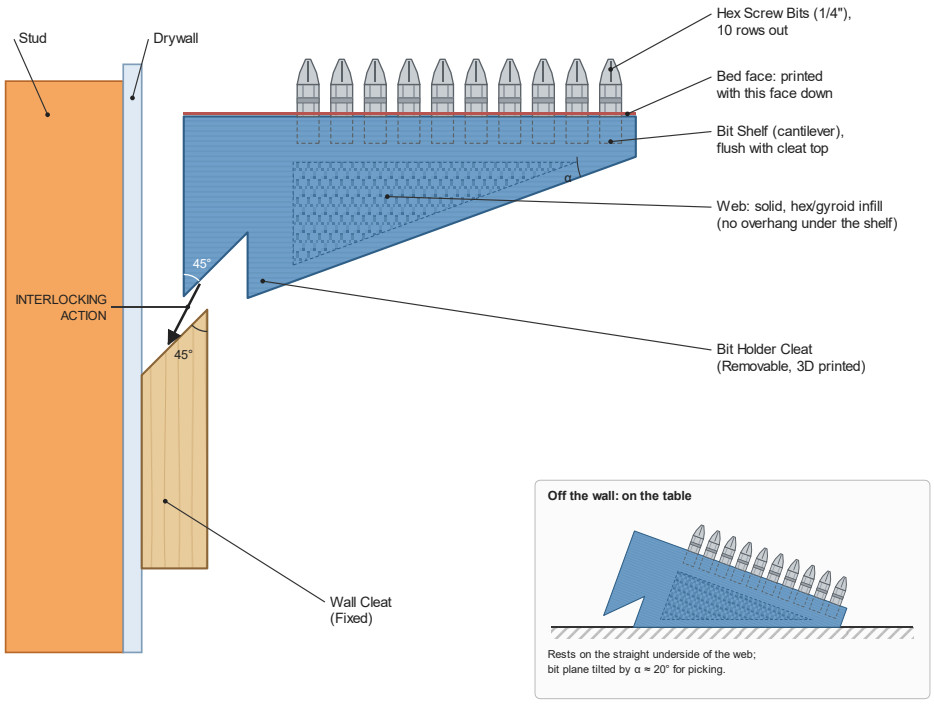


Figure 2–1: A screw bit holder for a French cleat (principle)

The example is a printed holder for 1/4″ (6.35 mm) hex screw bits that hangs on such a cleat. Bits sit in rows across the wall and rows outwards from it.

- Function: store bits within reach; to change bits, lift the holder off the cleat and use it at the bench.
- Interfaces: the wall cleat (45°, 18 mm plywood); 1/4″ hex bit shanks (6.35 mm across flats); quick-change bit holders with the same hex shank.
- Environment: workshop, basement or garage, e.g. 0–40 °C, dust, no direct sun.
- May vary: the number of bits across the wall and the number of rows outwards.
- Fixed: 25 mm bits, one row of quick-change holders next to the wall, PETG.
- Must stay true: the holder lifts off and goes back on without tools; bits stay in when the holder is carried or tilted; the cleat lip carries the loaded holder over time without creeping off.
- Not for: inserting or removing bits while the holder hangs on the wall; storing other tools.
- Ergonomics: the bits tilt 15° towards the user, for easy viewing and picking. The angle is fixed by the creator.
- Manufacturing: FDM, one piece, no supports. Every choice and calculation should stay within the build envelope of the available 3D printer.
- Open issues: the print orientation of the hook is not yet resolved; infill percentage.

© 2026 Magimo

2.2 Define the creator–user boundary

The design boundary defines which responsibilities remain with the creator and which may be delegated to the user. These responsibilities may be carried out by the same person, but the distinction remains useful when designing the model and its interface.

The creator is responsible for:

- defining the intended function and use conditions;
- structuring the model so that it remains stable within the supported range;
- identifying which inputs the user may provide or change;
- defining valid ranges and permitted combinations;
- encoding relevant relationships and manufacturing constraints;
- selecting appropriate defaults;
- testing representative and boundary configurations; and
- communicating manufacturing requirements, limitations and known failure modes.

The user is responsible for:

- providing accurate dimensions and application information;
- selecting from the permitted options;
- supplying accurate manufacturing context, where the workflow requires it, such as material, printer and orientation;
- using only the supported values and combinations, and using the part only under the stated conditions of use; and
- checking the printed part before use, or making sure the manufacturer does.

The user should not be expected to reconstruct hidden design logic. They should not have to decide whether a wall thickness is structurally adequate, calculate the correct clearance from a fit class, repair a broken sketch or guess which print orientation preserves the intended load path. If using the model requires this, the creator has transferred unfinished engineering work to the user.

The boundary is not determined only by the user's technical ability. An experienced engineer may be able to edit the Spreadsheet, change a sketch or override a value. That does not mean that these actions belong in the intended user interface. The relevant question is what the creator is prepared to support and take responsibility for within the defined design basis. This gives three kinds of information: context, choice and implementation:

- Context describes facts supplied by the user, such as the diameter of a mating component, the available installation space or the selected material.
- Choice describes an intentional variation, such as a fit class, overall length or approved edge style.
- Implementation describes how the model achieves the result. It includes offsets, construction dimensions, wall-thickness rules, transition lengths and feature-placement logic. Implementation covers the engineering and derived parameters of §2.4.

Context and meaningful choices may be exposed. Implementation should normally remain under the creator's control. It belongs in the expressions, geometry and protected engineering parameters that implement the design intent. The responsibility boundary should also be visible when the model is distributed. Technical skill does not move the boundary. An experienced

© 2026 Magimo

engineer can edit the Spreadsheet or a sketch, but that does not make it part of the user interface.

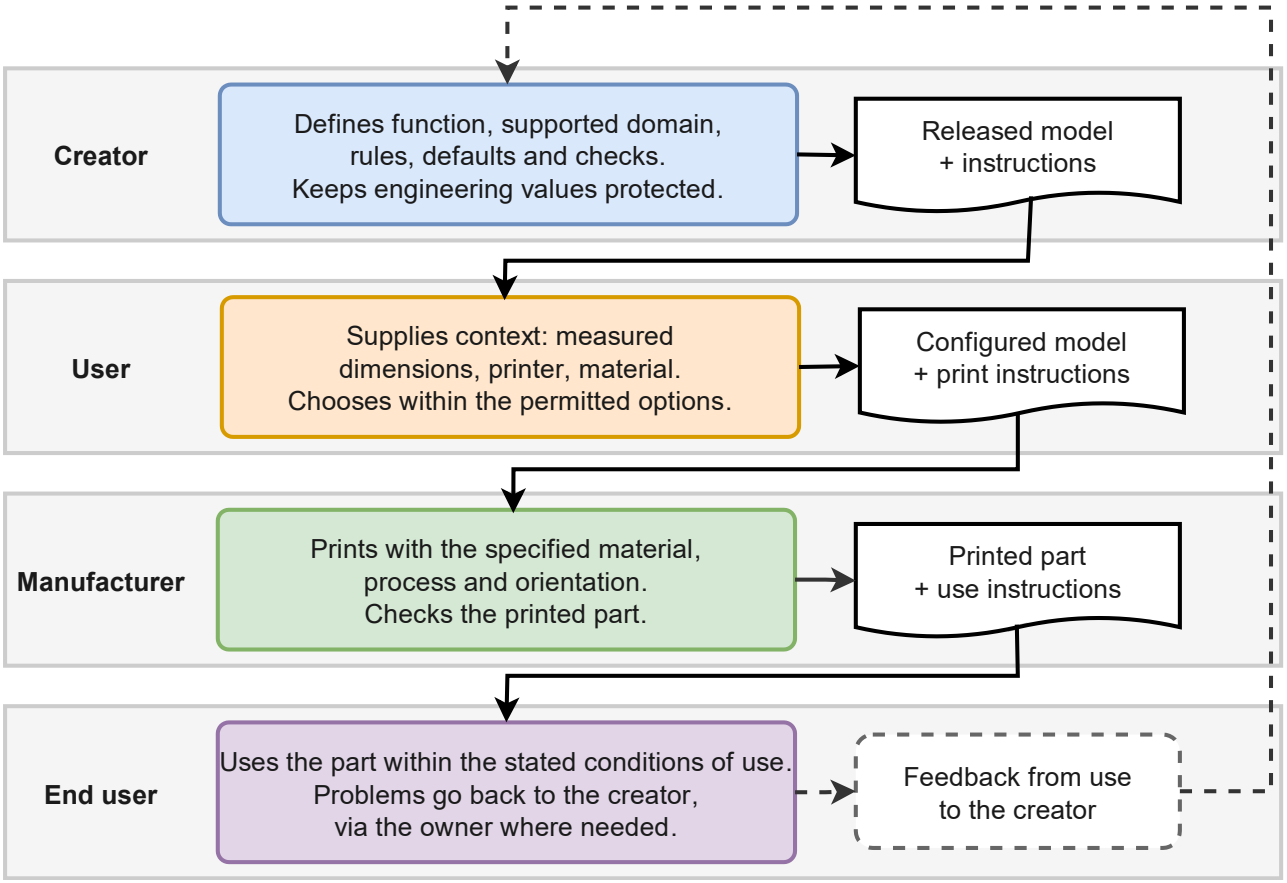


Figure 2–2: Responsibilities of the four roles. Each role supplies or checks what is within its responsibility and hands a defined result to the next. Feedback from use[3] returns to the creator, who decides on any change (see §6.5).

The boundary is what the creator supports and takes responsibility for. A user should be able to identify:

- which values they are expected to enter;
- which options are supported;
- which values are calculated;
- which manufacturing conditions are required; and
- which changes move the result outside the supported design.

This applies equally to a FreeCAD Spreadsheet, a scripted interface, a company library or a distribution platform. The interface may differ, but the responsibility boundary should not.

Bit holder

- **User:** decides how many bits across and how many rows outwards, and enters the build volume of their printer.

3 The end user may not be able to report problems, for example a child or a pet. The owner of the part then takes over that responsibility.

Bit holder

- **Creator:** keeps the material, the bit length, the socket clearance and spacing, the hook geometry, the load limits and the holder row.
- **Manufacturer:** prints in the stated material and orientation and checks the fit of the first bits.
- **End user:** lifts the holder off before changing bits; problems go back to the creator, via the owner where needed.

2.3 Define the supported configuration domain

A parametric model can often generate more configurations than its creator has designed, manufactured or tested. This is one of the most important distinctions in a distributed parametric system. A useful model distinguishes at least four domains:

- Generated configurations – configurations for which FreeCAD produces a result without an obvious recompute error.
- Geometry-checked configurations – configurations that do not contain known geometric contradictions such as self-intersections, zero-thickness regions or impossible feature combinations.
- Validated configurations – configurations that have been checked through appropriate analysis, printing, measurement, assembly or functional testing.
- Supported configurations – configurations that the creator is prepared to release with defined manufacturing conditions, use limitations and documented expectations.

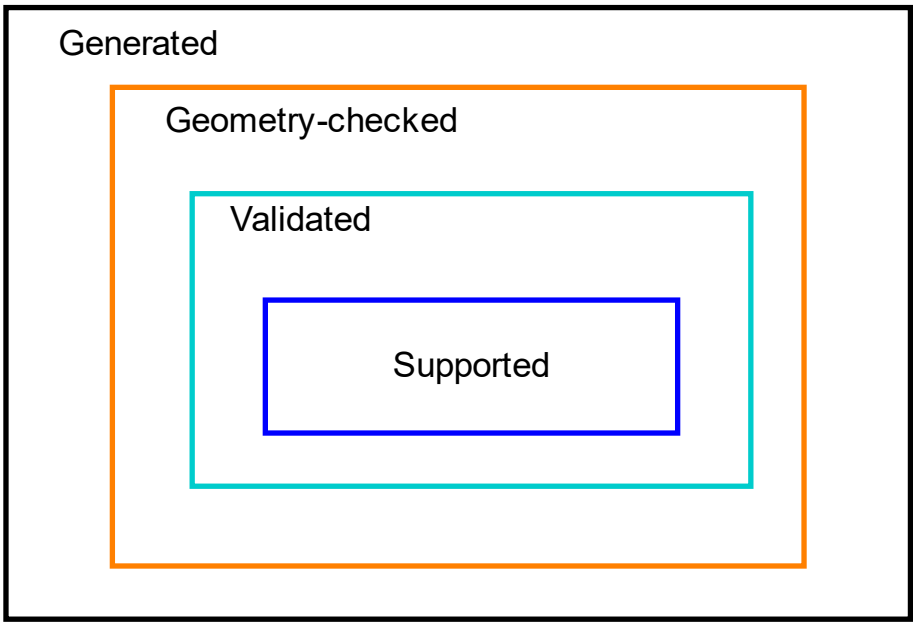


Figure 2–3: Increasing levels of assurance for a parametric configuration

A configuration may be generated without being geometrically admissible, validated or supported. The nested regions show increasing evidence and creator commitment; they do not imply that every configuration in an outer region is suitable for release.

The diagram is conceptual. These categories describe different questions and should not be treated as automatic or strictly nested status levels. Conversely, a configuration may have been tested during development but remain outside the supported domain because it has not been documented or released for users.

© 2026 Magimo

The supported configuration domain is therefore not simply the minimum and maximum value of each parameter. It also includes the relationships between parameters and the manufacturing and service conditions under which those relationships have been established. For example, a model may permit a length between 50 and 150 mm or a hole diameter between 5 and 30 mm. That does not prove that every combination within those ranges is suitable. At the shortest length, a thick wall may leave insufficient space for the hole. At the longest length, the same wall may be inadequate for the load. A large hole may also weaken a region that is acceptable with a smaller hole.

The valid domain may therefore be bounded by relationships rather than by independent limits. The creator should identify these relationships explicitly. They may include:

- minimum wall thickness as a function of nozzle and material;
- clearance as a function of fit type and process capability;
- maximum unsupported span as a function of orientation and layer height;
- rib spacing as a function of part length and expected load;
- feature count as a function of available surface or perimeter length; and
- allowable dimensions as a function of build volume.

Where a relationship cannot be represented reliably in the model, the configuration should be restricted, flagged for review or excluded from the supported domain. The default configuration should normally be both able to generate and supported. It should provide a known-good starting point rather than merely demonstrating that the model contains adjustable values. Additional configurations should be classified according to the evidence available for them. A practical release record should state:

- the supported parameter ranges and important dependencies;
- the applicable material and process assumptions;
- the required orientation and manufacturing conditions;
- the configurations tested and the evidence available;
- known unsupported combinations; and
- the response required when a requested configuration falls outside the declared domain.
- §7.5.1 lists the full release package.

For a distribution platform, this distinction can be represented directly in the configuration interface. A field may have a permitted range, but the platform should not imply that every value within that range has the same engineering status. Presets, status labels, warnings and manufacturing notes can help distinguish a tested configuration from one that is merely mathematically available.

> **Bit holder**
>
> The supported domain is bounded by relationships, not only by separate ranges:
>
> - Printer envelope: `rows_across` and `rows_out` may only go as far as the plate fits the usable build envelope in the released orientation.
> - Evidence: until the hook orientation is resolved and the creep test has passed, holders with more than e.g. 6 rows out are generated but not Supported.

A model's configuration domain is defined not by what the CAD system can generate, but by what the creator can justify, manufacture and support. Chapter 6 defines these terms in full.

2.4 Classify parameters by role

This section classifies parameter values, not people or job roles. A value is classified by what it represents in the design. The four parameter classes are context, choice, engineering and derived. Visibility, editability and dependency are separate properties and should not be confused with these classes.

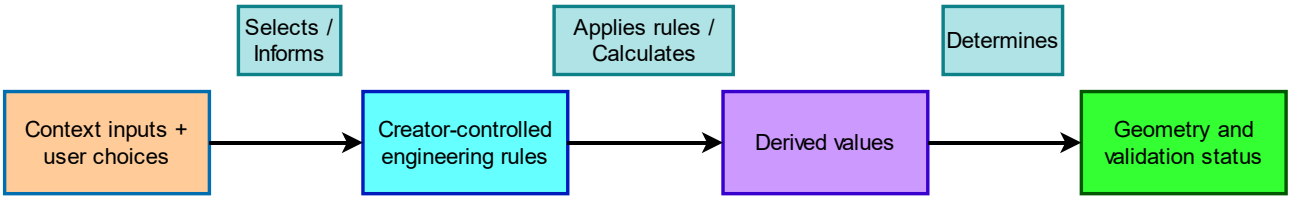


Figure 2–4: Conceptual view. Each inner level needs more evidence or commitment than the one around it.

An inner level needs the outer ones, but not the reverse. A configuration can recompute and still fail the geometry checks, or be tested and still not be supported.

Context and choice parameters provide the inputs; creator-controlled engineering parameters apply the design rules; derived parameters provide calculated values for the geometry. They do not, on their own, determine whether a configuration is supported.

The example below illustrates the flow from a mating diameter and selected fit class to a clearance rule and resulting hole diameter.

Context ---►	**Choice ---►**	**Engineering ---►**	**Derived**
Mating diameter; selected material or process profile	Fit class: loose, normal or tight	Approved clearance rule; minimum-wall rule	Calculated clearance; hole diameter

These classes describe the meaning of a value. They do not, by themselves, determine who controls it, whether it is visible to the user or whether it may be edited.

The same roles apply outside 3D printing. In a laser-cut finger-joint box, sheet thickness is context, box size is a choice, kerf compensation (correcting for the width of material the laser burns away) and minimum finger width are engineering rules, and the finger count is derived.

2.4.1 Context parameters

Context parameters describe the conditions in which the design is to be made, installed or used. They are facts about the application or manufacturing environment rather than design choices created by the model.

Some context values are supplied by the user. Others may be fixed by the creator or selected from a controlled manufacturing profile. A context parameter therefore does not automatically become a user-facing input. Examples include:

- hand width for a grip;
- cable or tube diameter;
- available installation space;
- printer build volume and usable XYZ limits;
- nozzle diameter and layer height;
- selected material;

© 2026 Magimo

- ambient temperature;
- relevant service conditions.

Material is normally a context parameter because it describes the manufacturing and service environment. It may also appear as a user choice when the creator permits selection from a controlled list of supported material profiles. The creator must define how each permitted material is treated by the design; materials may be required, restricted, indifferent or prohibited. See §2.5.

2.4.2 Choice parameters

Choice parameters represent deliberate product or configuration options. They describe something that the user is genuinely allowed to choose because the choice produces a meaningful and supported variation in the result. Examples include:

- fit class;
- overall length;
- approved pattern type;
- number of segments;
- mounting arrangement;
- orientation of a non-critical feature; and
- inclusion of an optional feature.

A choice parameter should not exist merely because the corresponding value can technically be changed in FreeCAD. It should represent a real choice with a predictable effect, a defined range or set of options and an appropriate level of validation.

2.4.3 Engineering parameters

Engineering parameters contain the values and rules used by the creator to make the design work. They may be fixed, protected or adjustable during development, but they are not normally delegated to the user. Examples include:

- minimum wall thickness;
- design margins and safety factors;
- transition length rule;
- process compensation factors;
- minimum feature spacing;
- support allowances;
- fillet and chamfer rules;
- material properties;
- physical constants; and
- protected design constants.

Material properties may be stored in a controlled material profile or referenced as part of the engineering basis. They should not become freely editable merely because they are available in a Spreadsheet.

© 2026 Magimo

Engineering parameters may be parametrised extensively. Internal parametrisation can make the model easier to maintain, optimise, test and automate. It does not mean that the user should be given control over these values.

2.4.4 Derived parameters

Derived parameters are calculated from context, choices, engineering values, material data and other relationships. They should normally have one authoritative definition and should not be entered independently in several locations. Examples include:

- resulting wall thickness;
- inner and outer diameters;
- calculated fit clearance;
- compensated hole diameter;
- rib spacing;
- resulting transition length;
- resulting fillet radius;
- feature count;
- centre of gravity.

A derived value may drive downstream geometry, but it remains dependent in origin. For example, a calculated fit clearance may depend on the selected fit class, material, nozzle and process assumptions. The resulting inner diameter may then drive the surrounding geometry.

A derived value should be shown as a result rather than presented as a second independent input. If the user can edit both the source value and the derived value, the model may contain two competing definitions of the same design requirement.

2.4.5 Driving and dependent values

The four classes above describe the role of information. A second classification describes dependency:

- Context, choice and engineering values can all drive the model.
- Material properties may be retrieved from a selected material and then drive calculations.
- Derived values depend on other values and relationships.
- Validation outputs depend on the resulting geometry and the limits defined by the creator.

Driving does not mean user-controlled. A material property, minimum-wall rule or internal fillet radius may drive the geometry while remaining entirely under the creator's control. The control layer should make these distinctions visible by showing where information comes from, what it represents, who controls it, whether it may be changed and what depends on it.

Bit holder. Every value in the model has one role

Context ---►	**Choice ---►**	**Engineering ---►**	**Derived**
- bit_hex_size (6.35 mm) - cleat thickness (18 mm, fixed) - plate_angle (15°)	- rows_across - rows_out	- socket_clearance (normal fit) - min_wall - bit_spacing - socket_depth_ratio - qh_depth	- socket_af - pitch - plate_width, - x_front, - t_front, - m_total,

- bed_x, bed_y, bed_z (nominal build volume of the user's printer)		- hook_depth - allowable_moment	- moment_self

The material (PETG) and the cleat thickness (18 mm) are context fixed by the creator: they describe the manufacturing set-up and the wall system, and the user cannot change them. `bit_hex_size` is context, not a choice: the bits determine it, not the user's preference.

Classify a value by its role before deciding where it belongs in the model. A value may drive the design without being a user decision.

2.5 Expose decisions, not implementation

The public interface should expose only the decisions that the creator has deliberately delegated to the user. It should not expose every value that can technically be changed in FreeCAD.

The user may need to provide application dimensions, select an approved configuration, identify the available printer or select the material loaded into that printer. These inputs describe the application or manufacturing context. They do not give the user unrestricted control over the engineering logic that follows from them.

Material selection illustrates this distinction particularly well. Before printing, the user needs to select a material. However, the creator must define what that selection means for the design. A material may be:

- required, because the design was intended and validated specifically for that material;
- restricted, because the user may choose only from an approved list of materials or material families;
- indifferent, because the material does not affect the relevant design intent under the stated process and service conditions; or
- prohibited, because the material or the resulting use would be unsuitable.

For example, a funnel designed for water may print successfully in several materials. This does not establish that it is suitable for petrol. Chemical compatibility and service conditions may be outside the intended use. If the material is relevant to the design, the user may select it from the permitted options. The model can then retrieve controlled material properties and apply the appropriate internal rules. The user should not normally be allowed to edit the Young's modulus, thermal properties, shrinkage factor or process compensation values associated with that material. If the material is indifferent to the design intent, the model does not need to introduce unnecessary material-dependent calculations. The material may still be recorded for manufacturing and traceability, but it need not influence the geometry.

The creator may parametrise these values internally to tune the model and keep relationships consistent. They remain the creator's responsibility because they belong to the implementation and engineering basis of the design. The following rule applies:

> Expose the user's required context and deliberate choices. Keep the engineering response to those inputs under the creator's control.

Expose a value only when you have deliberately decided that the user may control it; use the checklist in §3.5.7.

This does not mean that internal parameters should be hidden from the creator or maintainer. They should be named, documented and traceable within the engineering layer of the model. Protected means that the user cannot casually alter them, not that the design logic is secret.

© 2026 Magimo

For a distribution platform this distinction is essential. The platform should not expose every Spreadsheet alias as a public field. The creator should define the public configuration interface, the permitted material options, the required and prohibited conditions, and the internal rules that translate those inputs into geometry.

A parametric model may contain many adjustable values. The user should see only the values that the creator has deliberately chosen to make part of the user's responsibility.

Bit holder

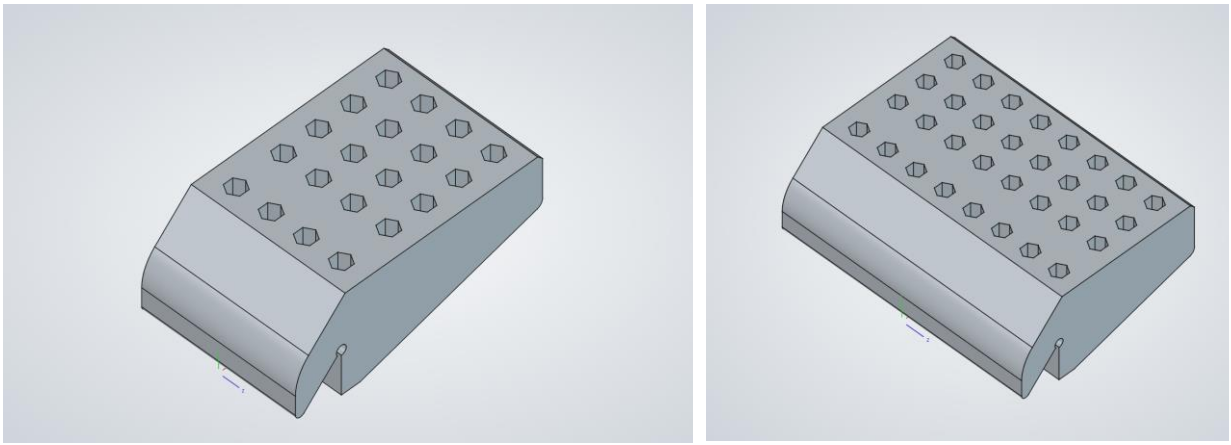


Figure 2–5: Two configurations of the bit holder: 4 × 4 and 8 × 3. The deeper holder row next to the wall takes quick-change holders or double-ended bits; it always spans the full width.

The holder row is not a user decision. Quick-change holders are the heaviest items, so a full row of deeper sockets sits nearest to the wall, and users often own more than one. It also suits double-ended bits, which are too long for the 8 mm bit sockets.

The creator fixes its depth (`qh_depth` = 20 mm) and its gap to the bit rows (`qh_gap` = 5 mm); the moment check includes its mass. The user sees two counts; the creator keeps the reasoning.

2.6 Design for explicit assumptions

Designing for explicit assumptions does not mean assuming that users know nothing. It means that the model must not depend on knowledge that exists only in the creator's head.

2.6.1 Counter the curse of knowledge

Engineers who work with additive manufacturing every day develop a vocabulary and a set of assumptions that can become invisible to them. This is called the 'curse of knowledge': once you know something, it is hard to see that others do not. This creates a particular risk when configurable products are supplied to experienced hobbyists, makers, small businesses or other users who did not create the model. Terms such as tolerance, clearance, anisotropy, bridging, support strategy and print orientation may be familiar to the creator but unclear to the person operating the model. Technically correct information may therefore fail to produce correct action.

The solution is not to remove technical language or hide engineering requirements. It is to explain the consequence at the point where the user must make a decision. Where necessary, give the technical term together with a plain description of what the user must do. Examples include:

- Instead of giving only a clearance value, explain what the fit is intended to do: “The joint is designed to assemble by hand and remain secure during normal use. Do not change the clearance manually.”
- Instead of giving only an anisotropy warning, state the required action: “Place Face A on the build plate. This orientation keeps the principal load in the stronger layer direction. Changing the orientation changes the strength assumptions.”
- Instead of assuming that support removal is obvious, identify the relevant surfaces: “Remove the temporary support material from the marked underside. Do not file or sand the mating surfaces unless the instructions specifically allow it.”

The technical term may still be useful for experienced users and future maintenance. It can be included in a description, tooltip, note or linked reference. The important point is that the user should not have to translate specialist terminology before knowing what to do.

2.6.2 What to assume about the user

A user may be technically competent and still lack the context in which the model was developed. They may not know why a particular print orientation is required, which material was used for validation, how a fit class is converted into clearance or why a seemingly harmless dimension must remain above a certain value. A distributed parametric model should therefore assume that the user:

- does not know the creator’s unstated design intent;
- does not know which parameters are related;
- does not know which values are implementation details;
- does not know which material or process assumptions are essential;
- may enter values that are physically plausible but outside the supported domain; and
- may interpret a successful CAD result as evidence that the part is suitable for manufacture and use.

The model must respond to these conditions through its structure and interface. Important assumptions should be expressed through meaningful names, descriptions, defaults, ranges, relationships and validation messages. A user should not need to inspect sketches, trace expressions or read source code to discover that two inputs are incompatible. For example, a user may enter a hand width that is realistic for a person but too large for the selected grip geometry. A printer may have sufficient nominal build volume but insufficient usable space once orientation, supports, adhesion features or machine limits are considered. A material may produce a valid solid in FreeCAD but be unsuitable for the intended chemical environment. These conditions should not be left for the user to discover after printing.

Designing for explicit assumptions requires the creator to:

- define the supported range of each public input;
- identify important dependencies between inputs;
- derive related values instead of asking the user to maintain them manually;
- encode non-negotiable manufacturing and functional limits;
- provide safe and representative defaults;
- identify required, restricted, indifferent and prohibited materials;
- validate combinations of values, not only individual values;
- explain the consequence of important choices; and

© 2026 Magimo

- state what remains outside the supported design domain.

The model does not need to accept every conceivable input. A smaller domain that is clearly defined and protected is more reliable than a wide range that has not been designed or tested properly.

2.6.3 User interface

The language used in the interface should relate to the user's decision and the real application. Names such as `offset2, radiusB` or `wall_adjustment` require the user to understand the model's internal construction. Names such as `hand_width, mating_diameter` or `installation_clearance` relate to the user's situation. More on naming in §3.3.

The internal implementation may remain complex, but the public interface should express the decisions in the language of the application. A short description should explain what the user must enter, which unit applies and what effect the choice has on the result.

2.6.4 Manufacturing and use information

The same principle applies to manufacturing and use information. If orientation affects strength, the required orientation must be stated. If a material is prohibited for a particular application, that limitation must be communicated. If global scaling invalidates wall multiples or fit clearances, the model must warn against scaling or provide a controlled alternative.

The user should also be told which manufacturing conditions are part of the design basis. These may include the printer, nozzle, layer height, material, process profile, support strategy, build orientation and post-processing requirements. Chapter 7 lists the full manufacturing and use handover (§7.2, §7.3).

> **Bit holder**
>
> The creator knows the holder must come off the cleat before bits go in or are taken out; a user does not. The model states it where the user reads the result, as a use note on the Result sheet and in the release notes:
>
> "Lift the holder off the cleat before inserting or removing bits. The cleat is designed for the holder's weight, not for pushing bits in."
>
> The same assumption returns as foreseeable misuse in §7.3.4.

2.6.5 Practical test

You know things about your model that nobody else does, so you cannot reliably judge its interface by using it yourself. Test it in two steps:

1. Put the model aside for a week. Then configure it using only the interface and its notes, and write down every time you have to rely on what you remember.
2. Give it to people who did not build it. Do not help them; note where they hesitate, guess or ask a question.

Each of those points is missing information: add it to the interface or the notes, not to a verbal explanation. The model passes when the user enters only the values asked for and gets a clear OK, warning or rejection before printing.

3 Build the Parameter Interface

This chapter turns the design boundary into a parameter interface: where each value lives, what it is called, who may see or change it, and which defaults, limits and choices apply. It ends with a review of the interface before any geometry is connected to it.

3.1 The Spreadsheet as a control layer

In a parametric model, the Spreadsheet should act as a control layer between design intent and geometry. It should not be treated merely as a convenient place to store dimensions or as the user interface by default. The control layer defines which values exist, what those values mean, where they come from, who controls them, which values depend on them, which limits apply and how the resulting configuration is checked.

The Spreadsheet is one possible implementation of this control layer. In a local FreeCAD workflow, it may contain both the user configuration and the creator's internal engineering logic. In a distributed system, these functions may be separated into different files, data structures, applications or services.

The key principle is the separation of responsibilities:

> The user supplies the application information and makes the configuration choices that define the requested result. The creator-controlled system translates those inputs into geometry, applies the engineering logic and reports whether the result is supported.

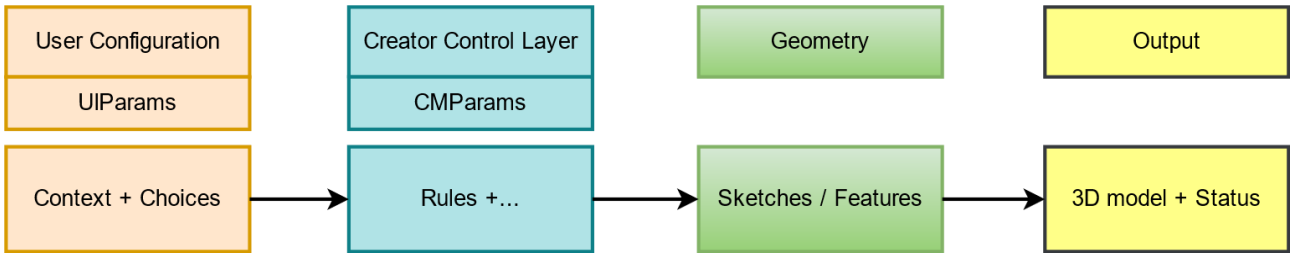


Figure 3–1: Spreadsheet control layer between user configuration and geometry

User context and choices enter the user sheet (UIParams); the creator sheet (CMParams) applies the rules, derives geometry-driving values and evaluates validation checks. Appendix B uses these sheet names.

Sketches and features consume those controlled values, producing geometry and a configuration status. A technically capable user may still inspect or edit the complete file unless the model-generation process is kept separate from the delivered configuration interface.

3.1.1 The user configuration layer

The user configuration layer contains the information and choices presented to the user for configuring the product. It is the public interface of the parametric product. It may contain user selections, application dimensions, user-defined boundaries, a selected printer or manufacturing profile, a selected material from the materials supported by the design, relevant environmental context, available feature choices and product-level preferences.

Parameters such as `hand_width`, `mating_diameter`, `installation_length` or `fit_class` represent requirements or choices that the creator has deliberately allowed to vary. They are not

arbitrary CAD dimensions. The creator must define the supported combinations and the evidence needed to justify them.

The user configuration layer should not expose the internal implementation of the design. It should not normally contain editable fillet radii, construction offsets, minimum wall rules, safety factors, material-property values, optimisation factors or other values that belong to the creator's engineering decisions.

A separate user-configuration Spreadsheet may be useful in a local FreeCAD workflow. It gives the user a clear place to enter application information and make configuration choices without exposing the creator's engineering logic. The user-configuration Spreadsheet can contain the values entered or selected by the user, together with short descriptions, units, defaults, available choices and status information. The creator-controlled Spreadsheet can then interpret these values, apply the engineering rules, calculate derived values and pass the results to the geometry. This separation improves readability and reduces the risk of accidental changes to protected values. It also makes the model easier to review and provides a clear boundary between user-supplied information and creator-maintained logic.

The user may select or load a material before printing, but not every material has the same status for the design. The creator classifies each material as required, restricted, indifferent or prohibited (§2.5). The creator also controls the material definitions, their properties, the process assumptions and what each selection changes in the design.

3.1.2 The creator control layer

The creator control layer contains the engineering logic that turns the user's configuration into a controlled design. It is maintained by the creator or maintainer and should not be presented as a general-purpose user interface. It may contain material-profile data, physical and design constants, process assumptions, minimum-wall and feature rules, design margins, clearance and compensation rules, optimisation parameters, feature-transition logic, protected geometry drivers, derived values, validation calculations and diagnostic states.

The creator may parametrise this layer extensively. Internal parametrisation can make the design easier to maintain, optimise, test and automate. It may also allow the same model to support several defined materials, printers or product variants.

Internal parametrisation does not mean that the user should be allowed to edit the underlying engineering values. A material selected in the user configuration layer may be passed to the creator control layer. The creator control layer can then retrieve the associated material properties, process assumptions and restrictions. It applies the appropriate rules and returns a supported configuration, warning or rejection. The dependency can be represented as:

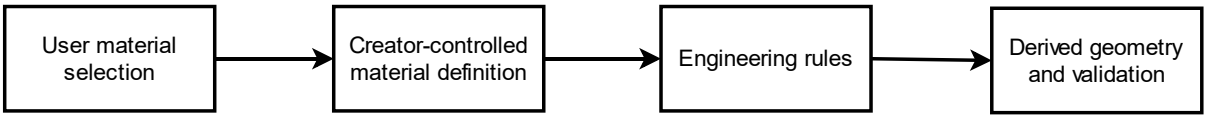


Figure 3–2: The user chooses the material or configuration option presented by the system. The creator controls the meaning and consequences of that choice.

3.1.3 Separation from geometry

The geometry should accept values from the creator control layer, not from arbitrary user edits. A typical arrangement is:

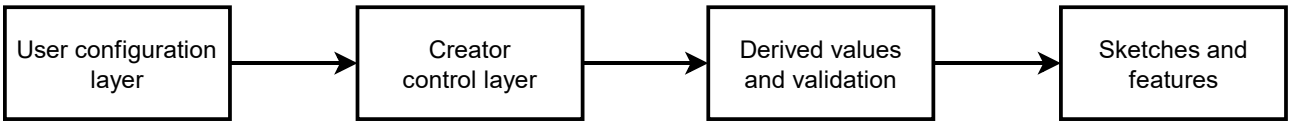


Figure 3–3: The user layer supplies application context and configuration choices. The creator layer interprets user inputs, applies the engineering rules and generates the controlled values required by the geometry.

The geometry should not need to know whether a value originated from a user input, a material lookup, a physical constant or a design rule. It should consume the appropriate controlled result. This follows the parameter roles in §2.4.

3.1.4 Possible implementations

The separation between the two layers can be implemented in several ways:

- separate user and creator Spreadsheets;
- separate sections within a controlled Spreadsheet;
- a public parameter file connected to a protected engineering file;
- a user-facing configuration form connected to a controlled CAD model;
- a controlled API that accepts defined fields; or
- a web interface connected to a controlled CAD engine.

The first two work inside FreeCAD today. The others need an application outside FreeCAD, such as a script, a web form or a platform (Chapter 7).

The physical arrangement is less important than the separation of responsibilities, data flow and access. Two Spreadsheets are useful in a local FreeCAD workflow because they make the distinction visible and help organise the model. However, separate Spreadsheets alone do not establish security. If the complete FreeCAD file is delivered, a technically capable user may still inspect or modify its contents.

Real separation needs two conditions: the user can reach only the configuration interface, and the creator keeps the logic that generates the model. A distributed FreeCAD file may provide organisation and guidance, but it should not be described as secure merely because it contains separate Spreadsheets. The external interface should not bypass the creator control layer by writing arbitrary values directly into sketches, features or internal engineering parameters.

A platform may therefore pass the user's choices and application information to a protected system that generates the model. That system applies the creator's parameters, material definitions, expressions, geometry rules and validation checks. It then returns the generated result and its status without exposing the internal model structure.

This separation also makes the model portable. The same creator-controlled engineering logic can serve a local FreeCAD user, an internal company tool, a print service or a distribution platform. The interface may change, but the meaning of the parameters, their limits, their dependencies and the design responsibilities stay the same.

Inside the .FCStd file, Spreadsheet cells are stored as XML, so scripts can read them. To exchange parameters with other systems, export the parameter schema to JSON or CSV (§B.9, §7.5.2). Other containers inside FreeCAD are compared in §3.1.5.

3.1.5 Alternative parameter containers in FreeCAD

Spreadsheets are not the only way to store parameters in FreeCAD. Object properties, parameter-set mechanisms and data-container tools may also be useful, depending on the FreeCAD version, installed workbenches and intended workflow. Each mechanism has advantages and limitations.

Table 3–1: Alternative parameter containers

Parameter mechanism	What it is	Pros	Cons	Use case
Parameter sets, such as VarSet (built in since FreeCAD 1.0)	Named, typed values that can be accessed by expressions or scripts.	- Structured and typed - less prone to accidental layout changes than a Spreadsheet - useful for scripted workflows; can be grouped logically	Less convenient for human-readable tabular documentation; availability and behaviour may depend on the FreeCAD version; may be less familiar to users	Internal engineering parameters and derived values rather than the primary user-facing configuration layer (see Appendix F).
Properties tab (object and document properties)	Every object and the document itself have a Properties tab that can store custom properties. These can be used as parameters and referenced in expressions.	- tightly coupled to the object or document; - visible in the standard GUI; - can be typed and grouped; - accessible from Python and expressions.	- scattered across many objects if not carefully organised; - easy to create inconsistent naming conventions; - less suitable as a single “control panel” for the whole model; - documentation and limits must be maintained separately.	local parameters tied to specific Bodies or features or document-level metadata (version, author, profile identifiers). See Appendix F.
Dynamic Data (add-on workbench, installed through the Addon Manager)	Objects or workbench features that store structured datasets, tables or lists.	- can store more complex structures than simple key-value pairs - useful for advanced data-driven workflows - can be linked to scripts and external tools	- needs the add-on on every machine that opens the file - may require additional workbenches or tooling - steeper learning curve; behaviour may vary across the FreeCAD ecosystem - often unnecessary for a basic control layer	Advanced material libraries, configuration tables or structured data workflows.
Configuration Table	A Spreadsheet table bound to an object, offering the user a drop-down list of defined configurations.	- native - well documented	the selector is an object property, not a Spreadsheet cell.	Discrete choices such as fit class and a supplier database (See Appendix C).

Practical recommendation

For the intended audience of this guide, a Spreadsheet-based control layer is often the clearest default because it combines visibility, documentation, compatibility with expressions and scripts and ease of review. Other mechanisms may be preferable for typed internal data, object-local

properties, metadata or headless automation. The choice should follow the required data structure, user interface, maintainability and FreeCAD version.

Bit holder

UIParams holds five values: the build volume of the printer (`bed_x, bed_y, bed_z`) and two counts (`rows_across, rows_out`).

CMParams holds everything else: the fixed context, the print basis, clearances, walls, hook geometry, the load limits, the derived dimensions and the checks. A separate Result sheet reports the status, size and mass to the user.

The sketches and patterns read only from CMParams; nothing in the geometry refers to UIParams directly.

	A	B	C	D	E	F
1	**Parameter ID**	**Display name**	**Value**	**Unit**	**Input type**	**Supported range / options**
2	**Printer used ...**					
3	bed_x	Build volume X	**220,00 mm**	mm	Numeric	100 mm to 600 mm
4	bed_y	Build volume Y	**220,00 mm**	mm	Numeric	100 mm to 600 mm
5	bed_z	Build volume Z	**250,00 mm**	mm	Numeric	100 mm to 600 mm
6	**Customised holder**					
7	rows_across	Bits across the wall	**4**		Integer	1 to max_rows_across (see Result)
8	rows_out	Bit rows from the wall	**4**		Integer	1 to 10 (above 6: review)
9						

	B	C
	Display name	**Value**
	Configuration status	**OK: configuration supported**
	Maximum # of bits across for your printer	14
	Maximum # of quick-change holders / double-end bits	16
	Width	61,98 mm
	Depth from the wall	99,61 mm
	Height	40,00 mm
	Total mass, filled	349,69 g

Figure 3–4: The spreadsheets UIParams (left) and parts of the Result (right) for the default 4 × 4. The user enters five values; the Result sheet reports the status, maximum number of bit positions, size and mass.

3.2 One source of truth

Each independent design value should have one authoritative definition. Other parts of the model should refer to that definition rather than storing separate copies.

This principle applies to both the user configuration layer and the creator control layer. The user interface may be the source of the information supplied by the user, while the creator control layer may contain the controlled internal representation used by the model. These are two stages in the data flow, not two competing design definitions. Typical examples:

- diameter or radius;
- material once;
- fit rule once;
- principal length once.

For example, the user may enter a mating diameter through a configuration interface. The model-generation system may transfer that value into the creator control layer. The transfer should be explicit and traceable. All subsequent geometry should refer to the creator-controlled representation rather than to several manually entered copies.

The same principle applies to calculated values. If the inner diameter is calculated from a mating diameter and a clearance rule, that calculation should exist in one authoritative location. Downstream sketches and features should refer to the calculated result rather than repeat the formula.

Duplicated values create a hidden risk. Two cells may initially contain the same number but no longer agree after a later change. The model may then contain two different versions of what the

creator intended, with no obvious indication of which one is correct. A common example is a user input for diameter combined with a separately entered radius. If the user changes the diameter but the radius is not updated, different features may be built around different interpretations of the same requirement. The correct arrangement is to choose one independent value and derive the other.

The same rule applies to constants and engineering rules. Avoid embedding unexplained values directly in sketches, expressions or feature properties. If a value affects the design, it should have a named location and a documented role. This applies even when the value is not exposed to the user. A creator-controlled model may therefore contain:

- one authoritative source for each user input;
- one controlled definition for each material or manufacturing profile;
- one location for each physical or design constant;
- one expression for each important engineering relationship;
- one calculated result for each derived parameter; and
- one validation rule for each supported-domain limit.

This does not require every value to be stored in one Spreadsheet. A large model may use separate Spreadsheets for user configuration, material data, engineering logic, derived values and validation. The requirement is that each value has one authoritative definition and that the direction of dependency between the layers is clear.

The rule also applies to external systems. A configuration record, a CSV file, a JSON object and a FreeCAD Spreadsheet should not become independent competing sources of design intent. One system should be identified as authoritative for each type of information, and the transfer between systems should be defined and traceable. If a value must be overridden during development, make the override explicit, temporary and documented. Do not allow an override to become an undocumented second definition of the design rule.

> **Bit holder**
>
> The user enters counts, not a plate width. `plate_width` and `x_front` (the depth from the wall) follow from the counts and the pitch. Offering both a width and a count would give two definitions of the same size. pitch is defined once in CMParams and used by both patterns, the plate outline and the moment calculation.

Chapter 7 shows how to name the authoritative source and transfer values to the others (§7.4.2, §7.5.2). A practical review question is:

If this value changes, where is the one place that must be changed to update the entire model correctly?

If the answer is more than one place, the model probably contains duplicate sources of reference. Define independent values once. Derive related values and refer to them everywhere else.

3.3 Names, aliases, units and meaning

Naming is part of the engineering structure of a parametric model. A name should communicate what a value represents, not merely where it happens to be stored or which feature currently uses it.

3.3.1 Names and aliases

Names such as `d_1`, `radiusA` or `cell_E14` force the reader to reconstruct the meaning from the model. They may be quick to create, but they become a burden when the model is reviewed, modified, automated or reused several years later. Prefer names that describe the physical or functional role of the value, such as:

- `thumb_width`
- `mating_diameter`
- `fit_clearance`
- `wall_thickness`
- `mounting_pitch`
- `nozzle_diameter`
- `layer_height`
- `ambient_temperature`
- `required_material`

The name should answer the question “What does this value mean?” It should not merely answer “Where is this value used?” An alias in FreeCAD should therefore be treated as part of the model’s vocabulary. An expression such as `mating_diameter + fit_clearance` is easier to understand and maintain than an expression based on cell references such as `B12 + F7`.

Named aliases make it easier to transfer a parameter structure to an external application, script or platform. An alias may be used as an external identifier, but only if the creator deliberately includes it in the interface contract and preserves its meaning across revisions. The displayed label can then be written in more accessible language. For example:

- internal identifier: mating_diameter;
- displayed label: Diameter of mating part;
- unit: millimetres;
- description: Enter the measured outside diameter of the component that must fit into the part; and
- supported range: defined by the supported design domain.

User-facing names and internal names do not always need to be identical. The important requirement is that the mapping between them is explicit and stable. If an alias is already part of a released model or external interface, changing it may be a compatibility change rather than a cosmetic edit.

Names should also distinguish similar but different quantities. For example, `overall_length` should not be used for `usable_length` and `outer_diameter` should not be used for `mating_diameter`.

Give each parameter one meaning. If a value would mean different things depending on another choice, use separate parameters. Example: if size means the diameter of a round hole but the width of a square one, use `hole_diameter` and `hole_width`.

3.3.2 Units

Units are part of the parameter definition. A value without a unit is incomplete information. The model should use a consistent internal unit system and should identify the expected unit for

every user-facing, engineering and derived value. For most FDM models, millimetres are a practical canonical unit for geometry, but the chosen system should be stated rather than assumed. Units should be defined for:

- dimensions;
- angles;
- mass;
- force;
- pressure;
- temperature;
- time;
- material properties;
- thermal properties; and
- environmental conditions.

Where FreeCAD permits values with explicit units, the model should handle them consistently. Where an external interface accepts only numerical values, the unit should be stated next to the field and included in the parameter schema.

Do not rely on regional settings, Spreadsheet conventions or personal assumptions. A value of 10 may mean 10 mm, 10 inches, 10 degrees or 10 Newton, depending on the parameter. The parameter name, description and interface must remove that ambiguity. Where the interface accepts only numerical values, the unit must be carried as explicit metadata.

3.3.3 Descriptions and documentation

Names are necessary but may not be sufficient. Important parameters should also have a concise description stating:

- what the value represents;
- where the value comes from;
- whether it is user-controlled or creator-controlled;
- what unit applies;
- what range or options are supported; and
- what effect a change has on the design.

For a user-facing parameter, the description should use the language of the application. For an internal parameter, it should explain the engineering reason for the value and identify the relationships that depend on it.

A material selector should identify the materials or material profiles supported by the design and indicate whether the selection affects clearance, wall thickness, strength assumptions, chemical compatibility or other design rules. The associated material properties should remain in the creator-controlled data layer and should not be confused with the user's selection.

Good names and descriptions reduce the amount of explanation that must be supplied elsewhere. They also make the model easier to audit, automate and hand over. A parameter name is not a label added after the engineering is complete. It is part of the model's interface and part of the design record.

© 2026 Magimo

Bit holder

Alias `rows_out`, label "Bit rows from the wall", input type Integer, range 1–10. Not n2 or rowsY: the name says which direction is meant, which matters because rows outwards change the load and rows across do not. `bit_hex_size` carries the description "shank size across flats", so nobody enters the across-corners value.

3.4 Group parameters by role and visibility

A parameter interface should be organised so that users can see what they need to provide or decide, while engineering logic remains protected and readable. A flat Spreadsheet containing inputs, constants, intermediate calculations and geometry dimensions in one sequence is difficult to audit and easy to misuse. Use the four parameter classes defined in §2.4 – context, choice, engineering and derived – to arrange the control layer. The classification describes what a value means; this section addresses where it should appear, who may see it and who may edit it.

The table in §3.4.1 groups parameters for interface purposes; "protected" and "calculated" describe control and dependency, not additional parameter classes.

3.4.1 Arrange by information flow

Keep Spreadsheet, VarSet, expression and alias structures organised, documented and consistent. Arrange the main Spreadsheet or control layer in the normal direction of information flow as defined in §2.4. A practical layout is:

Table 3–2: Practical layout for the information flow

Interface group	Typical content	Visibility	Editability
Context inputs	Printer envelope, nozzle, material, installation space, mating dimensions	Visible	Editable only where the user genuinely supplies the fact
User choices	Fit class, length, mounting variant, pattern, optional feature	Visible	Editable within approved options or ranges
Protected engineering values	Minimum wall, clearance rules, compensation, safety margins, process limits	Visible for auditability where appropriate	Creator or maintainer only
Calculated outputs	Effective wall, derived clearance, outer diameter, pattern pitch, status	Visible as results	Not independently editable

This order makes the responsibility boundary visible: context and choices enter the system, engineering rules process them, geometry consumes the controlled results, and calculated results are reported back to the user.

3.4.2 Separate user-facing and protected areas

Keep user-facing groups visually and logically distinct from protected values and calculated outputs.

- User-facing parameters should be grouped at the beginning of the interface or in a dedicated "UIParams" sheet. They should contain only values the intended user is expected to provide or select.

© 2026 Magimo

- Protected engineering values belong in a creator-control area or “CMParams” sheet. They may remain visible for review, but they should not be presented as ordinary editable inputs.
- Calculated outputs should be clearly identified as results. They may be shown to support transparency and diagnostics, but they must not be presented as a second way to enter the same design requirement.

3.4.3 Apply consistent editability

Editability is a separate decision from visibility. A value may be visible for auditability without being editable by the intended user. Apply the following default rules:

- Context inputs: editable when they describe a fact the user must supply; protected when they are fixed by the creator or supplied by a controlled profile.
- User choices: editable only within defined options, ranges or permitted combinations.
- Engineering values: protected from ordinary user editing; adjustable only by the creator or maintainer.
- Derived values: not independently editable; they must follow from their authoritative inputs and rules.

Do not rely on colour alone to communicate editability. Use clear group headings, parameter descriptions, status fields and, where available, controlled input mechanisms.

> **Making values read-only or hidden**
>
> Cells in the Spreadsheet cannot be locked. Keep protected values on the creator sheet (CMParams) and use a colour scheme (see §B.3.2) to make visually clear which cells the user may edit.

3.4.4 Review each parameter

For every parameter, ask:

- Does it describe the user’s reality, express a user choice, enforce an engineering rule or report a calculated result?
- Where does its authoritative definition live?
- Should the intended user see it?
- Should the intended user be able to edit it?
- Does its position in the interface reflect the information flow?
- Can a reviewer identify its source, role and dependency without inspecting sketches or expressions?

If a value does not fit one of the four roles, it may be redundant, poorly defined or located in the wrong layer. Decide the classification, placement, visibility and editability before connecting the parameter to geometry. The purpose is not only to make the Spreadsheet easier to read; it is to make the flow of information and responsibility clear to the creator, maintainer and user.

Bit holder

Interface group	Content	Visible	Editable
Context inputs	bed_x, bed_y, bed_z (the user's printer)	Yes	Yes, with unit
User choices	`rows_across, rows_out`	Yes	Within ranges and options

Protected engineering	`socket_clearance, min_wall, bit_spacing, qh_depth, hook_depth, allowable_moment`	Yes, for audit	Creator only
Calculated outputs	`plate_width, x_front, y_top, m_total, moment_self, config_status`	Yes, as results	No

`allowable_moment` stays visible so the user can see why a configuration is refused; it is never an input.

3.5 Defaults, limits and choices

A parameter interface must guide the user towards valid configurations. Each user-facing parameter should have a defined meaning, unit, default, supported range and, where appropriate, a restricted set of available choices. The objective is not to prevent useful variation. It is to prevent variation that the model cannot interpret reliably or that has not been validated.

3.5.1 Defaults are part of the design

A default value is not a placeholder. It is the first configuration most users will see and often the only configuration they will test. It should therefore represent a deliberate, useful and validated case. A good default should:

- produce a valid FreeCAD model without manual correction;
- remain within the declared manufacturing and service conditions;
- represent a realistic use case rather than an arbitrary midpoint;
- produce printable geometry under the intended process conditions;
- demonstrate the intended behaviour of the model; and
- give the user a sensible starting point for further adjustment.

For example, a default nozzle diameter should correspond to a supported manufacturing profile, not simply to the value most convenient for the geometry. The default configuration should be one of the tested configurations. If it has not been printed or otherwise checked, it should not be presented as though it were known to be reliable.

Defaults also communicate design intent. A default wall thickness, length, fit class or pattern selection tells the user what the creator considers normal. If the first configuration produces an overbuilt, marginal or visually confusing result, the model teaches the wrong lesson before the user has changed anything.

3.5.2 Define limits explicitly

Every user-editable numerical parameter needs a defined supported range. At minimum, define:

- the minimum value;
- the maximum value;
- the recommended or nominal range;
- the unit;
- the resolution or step size, where relevant;
- the reason for the limit; and
- the behaviour when the limit is exceeded.

© 2026 Magimo

These values should not be inferred only from the geometry. If a wall thickness must remain above a manufacturing limit, that limit belongs in the engineering logic and documentation. If a mounting distance must remain within the available Body length, the relationship should be explicit rather than left for the solver to discover through a failed recompute.

Do not confuse an individual limit with validation. A value may be valid on its own but invalid when combined with another value. For example:

- a hole diameter may be valid on its own but too large for the surrounding wall;
- a length may be within its individual range but too short for the selected pattern;
- a clearance may be positive but incompatible with the selected material and printer; or
- user-supplied dimensions may be valid geometrically but larger than the printer envelope.

The model therefore needs both individual limits and relationship checks. Individual limits control single parameters. Relationship checks control combinations. Where possible, reject invalid input before it reaches the geometry. If the model cannot reject it directly, use a clear diagnostic value or warning state. In FreeCAD, a status cell driven by an expression can report the problem to the user, for example “Outside supported range” (examples: §C.9.9, §E.3.8).

Do not silently replace a user’s value with another value unless the correction is deliberate, documented and unlikely to change the intended function. Silently changing a requested wall thickness from 0.8 mm to 1.2 mm may protect printability, but it can also alter weight, stiffness, internal clearance and fit. In such a case, the model should either report the enforced value clearly or require the user to correct the input.

3.5.3 Use discrete choices where they fit

Not every decision should be represented by a continuous number. Many design decisions are categorical:

- fit class: loose, normal or tight;
- edge treatment: sharp, rounded or chamfered;
- pattern: solid, lattice A or lattice B;
- support strategy: none, limited or permitted;
- orientation: horizontal, vertical or creator-defined; and
- manufacturing profile: a defined material, printer, nozzle and process combination.

A discrete choice is useful when the user thinks in categories and the creator has defined the engineering meaning of each category. In FreeCAD, offer discrete choices through a Configuration Table drop-down or a VarSet enumeration property (Appendix C and Appendix F). The selected value can then map internally to one or more engineering parameters. For example, a user may understand “normal fit” more readily than a clearance of 0.25 mm. The model can map the selected fit class to a clearance calculated from nozzle diameter, material behaviour and the intended interface. The user selects the functional requirement; the model applies the implementation.

This approach reduces the number of arbitrary numerical decisions, limits the number of combinations that require testing and allows the creator to change the internal rule without changing the user-facing concept.

A discrete choice is not automatically safer than a numerical input. It is useful only when every option has been defined, implemented and tested. Do not offer options that are visually distinct but have no reliable engineering or manufacturing basis.

3.5.4 Keep continuous inputs when reality is continuous

Continuous inputs remain necessary for measurements that come from the physical situation. Examples include:

- hand width;
- cable diameter;
- installation length;
- shaft diameter;
- mounting-hole spacing; and
- available installation space.

These values should not be converted into artificial categories merely to simplify the interface. Instead, provide:

- a clearly defined unit;
- a sensible range;
- a suitable input resolution;
- any required rounding or snapping rule; and
- a clear indication of whether the entered value is nominal, measured or required.

The model may derive printable geometry from the continuous input. For example, a measured cable diameter can determine the nominal channel size, while the fit class and manufacturing profile determine the additional clearance. This preserves the physical meaning of the user's input without exposing every downstream dimension.

3.5.5 Do not use limits to conceal uncertainty

A narrow range can make a model appear robust without making it genuinely robust. A parameter range should cover only the intended use that the design has been reasoned through and, where needed, tested.

Keep the terms apart: the interface may make a value available and FreeCAD may generate the result, but that does not make it validated or supported (§2.3, §6.1).

These terms are not equivalent. A configuration may recompute successfully but remain outside the Validated or Supported design domain. The interface should not imply a level of assurance that the evidence does not support. A range should therefore be based on the design basis and available evidence, not only on the values that allow FreeCAD to recompute.

3.5.6 Record the parameter definition

For each exposed parameter, maintain a compact definition. These records are the parameter part of the technical contract (§1.4). The full record may be stored in a Spreadsheet, external data file or platform database. The main fields should define:

- the user-facing label and internal alias;
- what the value describes;
- whether it is continuous, discrete, Boolean or derived;
- the expected unit;
- the default configuration;

© 2026 Magimo

- the supported individual range;
- the step or resolution, if applicable;
- the parameters that affect its meaning;
- the status of the resulting configuration; and
- the short instruction or warning shown to the user.

An exposed parameter is incomplete until its meaning, boundaries and consequences are defined. Keeping this information with the parameter also makes the interface easier to audit, automate and transfer to another system.

3.5.7 Before exposing a parameter

Before exposing a parameter, ask:

- Is it a decision the user is expected to make?
- Is the value meaningful to the user?
- Does exposing it transfer an engineering decision to the user?
- Could it be derived from another input?
- Is the default useful and tested?
- Are its individual limits known?
- Are important combinations with other parameters understood?
- Is a continuous value genuinely required?
- Would a discrete choice better represent the design decision?
- Does the model explain what happens at or beyond the boundary?
- Has the resulting configuration been validated to the level claimed?

If these questions cannot be answered, the parameter is not ready for the user interface. Hide it, derive it, define it more precisely or postpone exposing it until its behaviour is understood.

> **Bit holder**
>
> - Default: 4 across × 4 out, plus the quick-change holder row. This is the first configuration to print and test.
> - Limits: `rows_across` 1 to `max_rows_across`, the largest count that fits the bed (14 for a 220 mm bed); `rows_out` 1–10. Both are whole numbers.
> - Not exposed: bit length and material. Standard insert bits are 25 mm long, and PETG is the only qualified material. Every extra choice multiplies the configurations to test, so they stay fixed until there is evidence for a second option.
> - No silent correction: if the holder has more rows out than the supported domain allows, the model does not quietly reduce them. It reports the problem and leaves the choice to the user. “Review: more than 6 bit rows from the wall is not yet supported.”

3.6 Progressive disclosure and advanced controls

Not every user needs access to every available decision. A well-designed parametric model provides a small basic interface for normal configuration and a separate advanced layer for users

who need additional supported control. This allows expert flexibility without making every configuration decision compulsory.

The basic interface should contain only the inputs required to describe the intended use and the most important user choices. It should allow a user to reach a valid and useful result without inspecting the Spreadsheet, changing expressions or understanding the internal feature tree. Typical basic inputs may include:

- required physical dimensions;
- material or manufacturing profile;
- fit class;
- mounting or connection variant;
- length or size option; and
- one or two clearly defined feature choices.

The advanced layer may provide additional control over supported behaviour, but it must remain part of the defined interface. Advanced does not mean unrestricted. Every advanced parameter still requires a clear meaning, a supported range, a default and a known effect on the resulting geometry.

3.6.1 Basic and advanced layers

A practical division is:

- Basic configuration: any user; physical context, essential dimensions, validated options.
- Advanced configuration: experienced user; secondary dimensions, optional features, controlled geometric preferences.
- Engineering logic: creator or maintainer; safety margins, minimum wall rules, transition lengths, internal offsets.
- Calculated results: read-only; derived dimensions, warnings, status values, diagnostic results.

An expert user may be capable of changing an internal value, but that does not mean that the creator should make the value user-adjustable. See §2.6. If changing a value can invalidate strength, printability, fit or model stability, it should remain protected (§3.4.2).

3.6.2 Keep complexity out of the basic path

A user should not have to understand the full design system to obtain a valid result. If the basic path requires explanations of internal offsets, construction geometry, expression syntax or feature dependencies, too much implementation detail has reached the interface.

The basic path should answer only the questions necessary to construct the part.

> Expose enough control to describe the intended configuration. Keep the remaining complexity in the model.

Bit holder

The basic layer has the UIParams values. An advanced layer could add `bit_spacing` (finger access, now a creator rule), `cleat_thickness` (12, 15, 18 or 22 mm), `bit_length` (25 or 50 mm) and later magnet pockets or label recesses. Each advanced option needs its own range, default and test before it appears. The hook geometry, clearances and load limits stay out of both layers. A fit-class choice would be a natural advanced option, once the coupon has validated more than one clearance.

3.7 Review the parameter interface

Before connecting the parameter interface to sketches, features or expressions, review the interface independently of the geometry. At this stage, the question is not whether the model produces the correct shape. The question is whether the parameter system is sufficiently clear and controlled to justify driving the geometry.

3.7.1 Interface review checklist

Use the table below as a checklist for reviewing interfaces.

Table 3–3: Interface review checklist

Design Aspect	Key Question	Pass	Revise	N/A
Design intent	Is the purpose of the model and the intended use clearly defined?	☐	☐	☐
Parameter classification	Has every parameter been classified as context, choice, engineering or derived?	☐	☐	☐
User responsibility	Is it clear which values the user must provide and which choices the user may make?	☐	☐	☐
Meaningful names	Does every exposed parameter have a name that describes its physical meaning or functional purpose?	☐	☐	☐
Single source of truth	Is each independent value defined once, with derived values calculated from it rather than entered again elsewhere?	☐	☐	☐
Units	Does every numerical parameter have an explicit unit and, where necessary, a stated precision or step size?	☐	☐	☐
Default configuration	Does the default produce a useful, valid and tested configuration?	☐	☐	☐
Individual limits	Does each numerical input have a defined supported range?	☐	☐	☐
Combination limits	Have important interactions between parameters been identified, such as a hole diameter relative to wall thickness or a pattern relative to available length?	☐	☐	☐
Input type	Is each parameter represented in the correct form: continuous value, discrete choice, Boolean option or calculated result?	☐	☐	☐
Visibility and protection	Are basic inputs, advanced controls, protected values and calculated outputs visibly separated?	☐	☐	☐
Advanced controls	Does each advanced control meet a clear user need and have a defined validation basis?	☐	☐	☐
Manufacturing context	Are printer, nozzle, material, layer-height, orientation and other relevant process assumptions represented or documented?	☐	☐	☐
Failure behaviour	Is the intended response defined when an input is invalid or when a combination falls outside the supported domain?	☐	☐	☐
Documentation	Can a user understand what to enter without opening sketches or tracing expressions?	☐	☐	☐

© 2026 Magimo

Design Aspect	Key Question	Pass	Revise	N/A
External consistency	If the model will be driven by another interface, can the same parameter names, units, ranges and rules be transferred without ambiguity?	☐	☐	☐

3.7.2 Do not connect geometry prematurely

If an exposed parameter has no defined range, default, unit or purpose, do not connect it to the geometry merely to see what happens. That approach usually produces a working prototype with an undefined interface. Resolve the parameter definition first. Otherwise, later changes to names, categories, units or dependencies will propagate unnecessarily through sketches, expressions, external interfaces and documentation.

The review is complete when the parameter table can be read independently of the model tree and still explain:

- what the user enters;
- what the user may choose;
- what the creator controls;
- what the model calculates;
- what configurations are supported; and
- what happens outside those limits.

This table is the interface part of the technical contract. Chapter 4 builds a FreeCAD model that keeps to it.

> **Bit holder**
>
> The review against the §3.7.1 checklist found two items to revise before any geometry was connected:
>
> 1. Inputs: `bit_length` and `material` were user choices; they became fixed context (25 mm, PETG), which made the configuration simpler and shrank the test matrix.
> 2. Units: `bed_x`, `bed_y` and `bed_z` accepted a bare number; the description now asks for a value with unit (e.g. 250 mm), so the envelope check cannot compare millimetres with plain numbers.
>
> Both were cheap to change in this stage and expensive after the sketches and patterns had been built.

4 Build a Robust FreeCAD Model

This chapter makes the FreeCAD model behave reliably across the whole supported domain, not just at its default values. It covers stable references, master sketches, constraints, expressions, feature-tree structure and performance, so that changing a parameter changes the part as intended and nothing else.

4.1 Robustness means reliable behaviour under change

A parametric model exists to be changed. If it only produces the intended result at one nominal set of values, it is not yet a parametric product. It is a conventional CAD model with additional inputs – and therefore additional ways to fail.

Robustness, in this guide, means that the model behaves reliably when parameters are changed within the supported configuration domain. A robust model:

- recomputes without broken sketches, missing references or inverted features;
- preserves the intended function and key relationships when dimensions vary;
- fails in a controlled and understandable way when limits are exceeded; and
- keeps the dependency structure comprehensible to a future maintainer.

4.1.1 A nominal model is not a tested model

Many models are created and tuned for one “golden” configuration. At that nominal point, everything fits, prints and functions as intended. That does not prove that the model is robust.

A parametric product must be tested across a range of configurations:

- minimum, nominal and maximum values for key dimensions;
- representative combinations of user choices;
- boundary cases where two or more parameters interact; and
- configurations that intentionally approach or slightly exceed the supported limits to observe the failure mode.

Only after such testing can the creator state with confidence which configurations are supported, which can merely be generated and which should be rejected or flagged for review.

4.1.2 Geometry alone is not enough

Robustness is not only a property of the geometry. It is also a property of the parameter architecture, reference strategy, constraint scheme, expression structure and feature tree. A model may have perfectly valid geometry at one configuration and still be fragile because:

- sketches reference transient topological elements that change when dimensions vary;
- constraints are over-defined or rely on implicit assumptions;
- expressions contain hard-coded cell references or duplicated constants;
- features depend on intermediate geometry that disappears under certain parameter combinations; or
- the dependency chain is so complex that no one can safely modify the model.

© 2026 Magimo

A robust parametric model is therefore a system in which geometry, parameters, constraints, expressions and feature order work together to preserve design intent under change.

4.1.3 Robustness has a defined domain

Robustness is not absolute. It is defined with respect to:

- the supported parameter ranges;
- the permitted combinations of choices;
- the manufacturing assumptions (printer, nozzle, material, orientation, process profile); and
- the intended service conditions.

A model can be robust within its declared domain and intentionally unsupported outside it. For example, a grip may be robust for hand widths between 70 and 110 mm, for three defined fit classes and for a specified material family. It is not a failure of the model that it does not behave sensibly at 30 mm hand width, with an untested material or with arbitrary global scaling.

The creator's task is to:

- define the domain in which the model is intended to work;
- design and test the model so that it is robust within that domain; and
- communicate the domain and its limitations clearly to the user.

4.1.4 Design for controlled failure

Not every invalid input can be prevented by limits alone. Users may select unusual combinations, ignore warnings or operate outside the stated conditions. A robust model should therefore be designed to fail in a controlled way:

- sketches should remain solvable or fail with clear diagnostics rather than collapsing into degenerate geometry;
- expressions should produce meaningful error states or warning flags instead of silently generating nonsense;
- features should either build correctly or fail early with an understandable message; and
- the model should make it obvious when a configuration is outside the supported domain.

Controlled failure is preferable to silent corruption. A user who receives a clear warning or a non-building feature is less likely to manufacture and use an unsafe or non-functional part than a user who sees a valid solid and assumes everything is fine.

4.1.5 Robustness is an engineering property

Robustness is not a cosmetic feature. It is an engineering property that affects safety, reliability and trust. A fragile parametric model can generate many configurations quickly, but if those configurations are not dependable, the model undermines the credibility of parametric design as a whole.

Investing in robustness – through careful references, clear constraints, disciplined expressions and systematic testing – is therefore part of the creator's professional responsibility. It is what separates a configurable engineering product from an experimental CAD exercise.

© 2026 Magimo

4.2 Stable references and the Topological Naming Problem

FreeCAD, like many CAD systems, can suffer from the “Topological Naming Problem”: when the shape or order of features changes, the internal names of faces, edges and vertices may also change. Sketches or features that reference those elements can then break or behave unexpectedly. A robust model minimises unjustified dependence on unstable topological references and uses deliberate references wherever practical.

> FreeCAD continues to improve its handling of topological naming. These improvements can reduce the frequency of broken references, but they do not make every face, edge or vertex reference semantically stable. A model should therefore be designed on the assumption that generated topology may change when dimensions, feature order, patterns, fillets or configuration-dependent features change. The robust response is not to rely on a particular internal edge or face name, but to create and use references that express the intended engineering relationship.

4.2.1 The failure mechanism

A typical failure occurs when a downstream sketch is attached directly to a generated face. At the time the sketch is created, the selected face may be identified internally as Face1, Face6 or another generated subelement. The sketch does not necessarily understand the engineering meaning of that face; it records a reference to the result produced by an earlier feature. If an upstream parameter changes the shape or feature sequence, the solid may be rebuilt with a different collection or ordering of faces and edges. The original face may then receive a different internal name, disappear or be replaced by a topologically similar but functionally different face. The downstream sketch can consequently break, detach or remain attached to the wrong location.

> **A simple topological naming failure**
>
> Consider a two-level block:
>
> 1. A base rectangle is sketched on the XY plane and padded.
>
> 2. A second sketch is created directly on the top face of that pad.
>
> 3. The second sketch is padded to create the upper level.
>
> 4. Later, a groove or chamfer is added to the base feature, or an upstream parameter changes the feature topology.
>
> The original top face may no longer be represented by the same internal subelement. The second sketch can then detach, fail to build or remain associated with an unintended face. In the last case, the model may appear valid while the upper level is no longer located according to the original design intent.
>
> A more robust arrangement is to place the second sketch on a deliberate reference plane and define its offset from the model’s parameter structure:
>
> - Create or use a stable datum or origin plane.
> - Define the vertical position through a parameter or expression, such as the base height.
> - Attach the second sketch to that plane rather than to the generated top face.
> - Test the relationship when the base height and upstream feature choices change.
>
> The result is not merely a workaround for a renamed face. It expresses the intended relationship directly: the upper level is located at the controlled base height.

The important point is that the *visible* shape may still look plausible. A model can recompute without a clear error while a hole, pocket, fillet or text feature has moved to an unintended face. Reference stability must therefore be checked by observing both recompute status and design intent.

4.2.2 Prefer deliberate references

Prefer references that express design intent rather than references selected only because they are convenient in the current model state. Suitable references may include origin geometry, datum planes, datum axes, master sketches, stable construction geometry, named features or deliberately selected faces and edges whose identity is tested within the supported domain.

Deliberate references make the model's intent explicit and reduce the risk that a parameter change will break downstream features.

4.2.3 Separate architectural and local references

It is useful to distinguish between:

- Architectural references – the main datums, axes, planes and master sketches that define the overall layout and load paths of the part. These should be few, stable and clearly identified.
- Local references – sketch constraints, construction lines and temporary datums used to resolve a specific feature. These should be kept local to their sketch or feature and should not be referenced by distant parts of the model.

Architectural references form the backbone of the model. Local references are implementation details that may change without affecting the overall design. Keeping them separate makes the model easier to understand and maintain.

4.2.4 Keep references semantically clear

A reference should communicate its purpose. For example:

- a datum plane named "XY_base" clearly indicates the primary build plane;
- an axis named "load_direction" indicates the principal load path;
- a master sketch named Skeleton_main indicates the core layout.

Avoid generic names such as "Plane001", "Axis002" or "Sketch005" for important references. Rename key datums and sketches to reflect their role in the design.

4.2.5 Test reference stability

Reference stability should be tested as part of the configuration testing described in Chapter 6. In particular:

- vary key dimensions across their range and observe whether sketches or features break;
- change user choices (e.g. pattern type, mounting variant) and check that references remain valid;
- suppress or reorder non-critical features and confirm that the core model still builds; and
- document any known fragile references and their supported conditions.

If a reference proves unstable under supported configurations, redesign the sketch or feature to use a more stable reference rather than restricting the parameter range to work around the problem.

© 2026 Magimo

For a model intended to be driven through a distribution platform, reference stability is part of the release contract. The model must be tested through the same parameter interface that users or automated services will use. Testing only the creator's nominal configuration is insufficient. A platform-oriented reference review should identify:

- sketches attached to generated faces, edges or vertices.
- External geometry taken from generated solid topology.
- Datum objects whose support is itself topologically fragile.
- Features that change, disappear or reorder across variants.
- References that remain technically valid but no longer express the intended design relationship.

> A reference is not robust merely because it works in the nominal configuration. Change the parameters that influence the reference, recompute the model and verify that the dependent features still refer to the intended geometry. Record any parameter combinations that are outside the supported domain rather than treating them as reference failures.

4.3 Skeletons, master sketches and functional reference geometry

A skeleton or master sketch is a small set of sketches, datums and reference geometry that captures the main relationships and layout of the part. It is the architectural core from which the rest of the model is derived.

A master sketch and stable datums define the part's functional layout; downstream sketches and features reference that architecture. Avoid making the model depend on arbitrary generated faces or edges that may change when the geometry is edited.

4.3.1 Master sketches define the main relationships

A well-designed master sketch typically includes:

- principal outlines and centre lines;
- key mounting points and interfaces;
- main load paths and symmetry planes;
- critical distances and angles that govern the design; and
- construction geometry that encodes the primary design rules.

Downstream sketches and features should reference the master sketch rather than reinventing these relationships. This ensures that changes to the core layout propagate consistently through the model.

4.3.2 Use a small number of purposeful references

A skeleton should be minimal and purposeful. It is not a place to dump every dimension and constraint. Include only the geometry and relationships that:

- define the overall shape and layout;
- control the main functional interfaces; and
- encode the key design rules that must remain true under change.

© 2026 Magimo

Additional details – such as local fillets, small holes or decorative features – should be defined in separate sketches or features that reference the skeleton, not embedded within it.

4.3.3 Establish a deliberate dependency direction

The dependency direction should be clear and consistent:

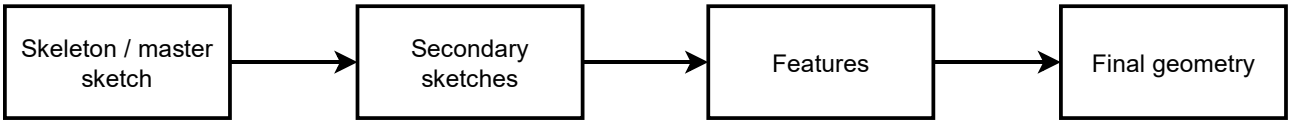


Figure 4–1: Deliberate dependency direction

Avoid circular dependencies where a downstream feature indirectly drives a dimension in the skeleton. Such loops make the model difficult to understand and prone to solver conflicts.

If a downstream feature needs to influence the skeleton (for example, an optimisation result), do this through explicit parameters or a separate optimisation layer, not by direct geometric feedback into the master sketch.

4.3.4 Reference function, not only shape

The skeleton should represent function, not just outline. For example:

- a line in the skeleton may represent the intended load path, not just an edge;
- a point may represent a mounting interface or pivot centre, not just a vertex;
- a plane may represent the build orientation or a critical assembly interface.

By referencing function, the skeleton remains meaningful even if the detailed shape changes. This makes the model more adaptable and easier to modify for new requirements.

4.3.5 Connect downstream geometry deliberately

Downstream sketches and features should connect to the skeleton in a controlled way:

- use external geometry references from the skeleton rather than picking arbitrary edges;
- constrain downstream sketches to skeleton geometry where it defines the layout;
- keep local details local – do not over-constrain a detail sketch to distant skeleton elements unless functionally required.

This approach keeps the model modular: the skeleton can be modified with predictable effects, and local features can be adjusted without destabilising the whole structure.

4.3.6 Do not confuse a skeleton with the finished part

The skeleton is an abstraction, not the final geometry. It may omit fillets, chamfers, small holes and other details that are added later. It may also use simplified shapes that are easier to constrain and understand.

Do not try to make the skeleton look like the finished part. Its purpose is to capture relationships, not to be a minimal version of the final geometry.

4.3.7 The architectural rule

A useful rule of thumb is:

> If a change in this parameter should affect the overall layout, it belongs in or near the skeleton. If it should only affect a local feature, it belongs in a downstream sketch or feature.

This helps keep the model organised and makes the effect of parameter changes more predictable.

Bit holder

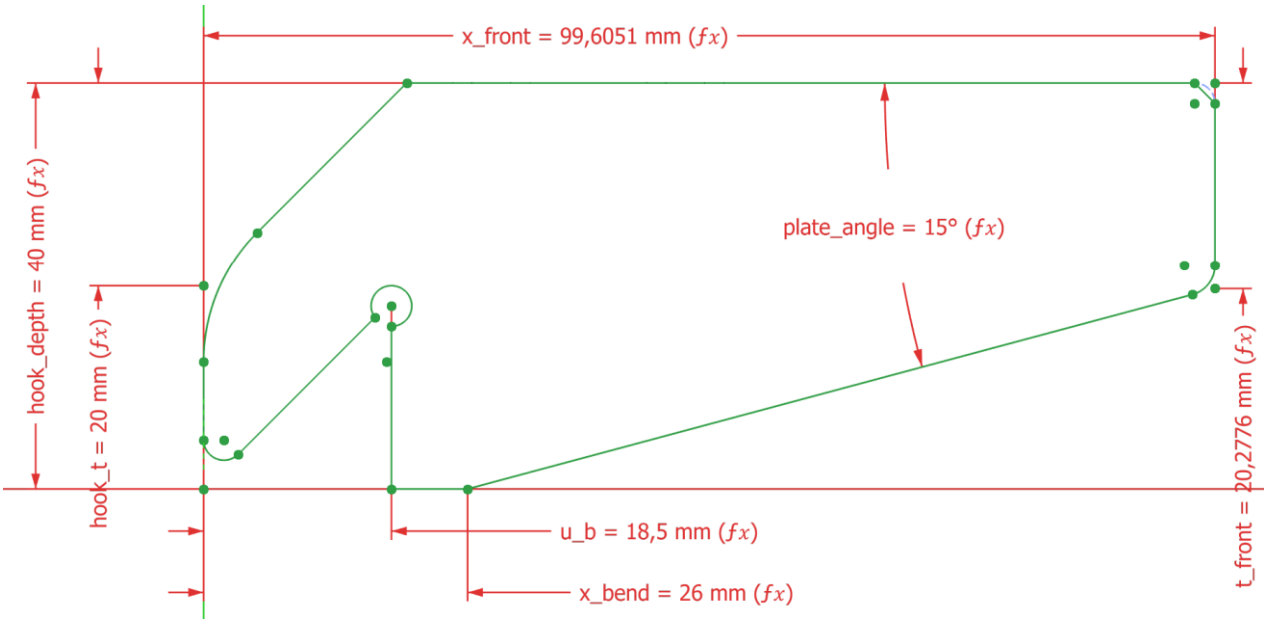


Figure 4–2: The profile sketch. Everything is measured from the wall; the hook shape hangs from the top corner, so it moves with the height.

The functional reference is the wall: x = 0 at the wall face, y = 0 at the bottom. The profile sketch on the side plane holds the rear face (`u_b`), the start of the slope (`x_bend`), the front face (`x_front`) and the front thickness (`t_front`). The hook shape hangs from the top-rear corner, so it follows when the height grows. The socket sketches measure from the same origin (`x_bits1`, `x_qh`). Changing `rows_out` moves the front face, never the wall face or the hook.

4.4 Constraint strategy

Constraints are the rules that define how sketch geometry behaves. A good constraint strategy ensures that sketches are fully defined, stable and aligned with the design intent.

4.4.1 Fully constrained, but not over-constrained

Functional production sketches should normally be fully constrained or otherwise deliberately controlled. Construction and exploratory sketches may be treated differently, but their role and status should be clear. However, they should not be over-constrained with redundant or conflicting constraints that make the solver's job harder and the sketch more fragile. Signs of over-constraining include:

- solver warnings about redundant constraints;
- constraints that fight each other when a dimension is changed;
- construction geometry that is unnecessarily locked; and
- a feeling that the sketch "barely solves" and breaks easily.

Aim for a minimal, clear set of constraints that fully defines the sketch without unnecessary locking.

4.4.2 Driving and driven dimensions

Distinguish between:

- **Driving** dimensions – parameters that control the sketch (e.g. overall length, hole spacing, wall thickness). These are typically linked to user inputs or engineering parameters.
- **Driven** dimensions – measurements that result from the geometry and driving dimensions (e.g. total perimeter, remaining material, derived angles). These should be shown as reference or driven dimensions, not as independent inputs.

Using driven dimensions correctly prevents conflicts where two dimensions try to control the same geometry.

4.4.3 Anchor geometry to functional references

sketches should be anchored to functional references, such as:

- origin or principal datum planes;
- master sketch geometry;
- named axes or points that represent interfaces or load paths.

Avoid anchoring sketches to arbitrary vertices or edges that may change when parameters vary. Functional anchors make the sketch's position and orientation meaningful and stable.

4.4.4 Keep each sketch comprehensible

Each sketch should be easy to understand at a glance:

- use construction lines and points to clarify intent;
- group related geometry logically;
- avoid excessive construction geometry that obscures the main shape; and
- name or comment complex constraints where the CAD system allows it.

A comprehensible sketch is easier to debug, modify and hand over to another engineer.

4.4.5 Avoid constraints that conceal the design intent

Constraints should express the design intent, not hide it. For example:

- use a symmetry constraint when symmetry is intended, rather than manually equalising two dimensions;
- use a tangency constraint where a smooth transition is required, rather than approximating it with many small segments;
- use an angle or parallel constraint where orientation is critical, rather than relying on implicit alignment.

Constraints that match the engineering intent make the model more robust and easier to justify during review.

4.4.6 Test the sketch as a system

A sketch is a small system of geometry and constraints. It should be tested as such:

- vary driving dimensions across their range and observe whether the sketch remains stable;
- check that the sketch does not flip, invert or degenerate at boundary values;
- confirm that the sketch behaviour matches the intended design logic; and
- document any known limitations or special conditions.

If a sketch behaves unpredictably under supported configurations, revise the constraints or references rather than restricting the parameters to avoid the problem.

4.5 Expressions: centralise logic, simplify geometry

Expressions are a powerful tool in FreeCAD. They allow relationships to be defined once and reused throughout the model. Used well, they make the model more maintainable and transparent. Used poorly, they create opaque dependency chains that are difficult to audit.

4.5.1 One relationship, one location

Each important relationship should be defined in one authoritative location. For example:

- a clearance rule should be defined once and then referenced by all relevant features;
- a wall-thickness formula should exist in one cell or expression, not repeated in multiple places;
- a material-dependent compensation factor should be retrieved from a single source.

This avoids the situation where a change requires updating multiple expressions, with the risk that some are missed.

4.5.2 Keep sketch expressions short

sketches should reference parameters and simple expressions, not embed complex logic. If a sketch constraint requires a complicated formula, define that formula in the Spreadsheet or a dedicated expression container, then reference the result in the sketch.

This keeps sketches readable and makes the logic easier to test and modify.

4.5.3 Keep geometric relationships in the sketch

While complex calculations should be externalised, purely geometric relationships that belong to the sketch should remain there. For example:

- symmetry, tangency and parallelism are best expressed as sketch constraints;
- local proportionalities (e.g. “this radius is half this width”) can often be handled cleanly within the sketch.

The guideline is:

- put engineering rules and calculations in the Spreadsheet or expression layer;
- put geometric relationships that define the shape in the sketch.

4.5.4 Avoid raw cell references and duplicated constants

Avoid expressions such as `Spreadsheet.B12` or `Spreadsheet.F7` unless absolutely necessary. Instead:

- assign meaningful aliases to key cells (e.g. `nozzle_diameter, fit_clearance`);

- use those aliases in expressions and sketches; and
- avoid hard-coding constants in multiple places.

This makes the model more robust to Spreadsheet reorganisation and easier to understand.

4.5.5 Avoid circular and opaque dependency chains

Expressions should form a clear, acyclic dependency graph that runs one way: inputs, engineering rules, derived values, geometry. Avoid circular dependencies where A depends on B, B depends on C and C depends back on A. FreeCAD may sometimes resolve such loops, but they make the model fragile and hard to debug.

Also avoid long, nested expressions that are difficult to read. Break complex formulas into intermediate steps with meaningful names.

4.5.6 Expressions should support testing

Expressions should be written so that their behaviour can be tested:

- use clear, named intermediate values rather than one giant formula;
- include diagnostic expressions where helpful, for instance:

```
wall_check = wall_thickness < min_wall ? <<FAIL>> : <<OK>>
```

- document non-obvious formulas with comments or adjacent description cells.

This makes it easier to verify that the model behaves correctly across the supported configuration domain.

> **Bit holder**
>
> The engineering logic sits in CMParams, one relationship per cell:
>
> - `socket_af = bit_hex_size + socket_clearance`
> - `socket_ac = socket_af / cos(30)`
> - `pitch = socket_ac + max(bit_spacing; min_wall)`
> - `plate_width = (rows_across - 1) * pitch + socket_ac + 2 * edge`
> - `x_front = x_bits1 + (rows_out - 1) * pitch + socket_ac / 2 + edge`
> - `t_front = max(socket_depth + floor_t; hook_depth - run * tan(plate_angle))`
> - `moment_self = 9.81 m/s^2 * (m_holder * (u_b + x_front) / 2 + m_bits * x_bits_c + m_qh * x_qh)`
> - `MomentPass = moment_self <= allowable_moment ? 1 : 0`
>
> Notes:

- `pitch` shows two rules competing: the wall between two sockets must be at least `min_wall` (1.8 mm), but fingers need `bit_spacing` (6.5 mm) to pick a bit out. max() makes the stricter rule win, and both rules stay visible: `pitch` = 7.6 + 6.5 = 14.1 mm.[4]
- `t_front` shows a clamp: the front is as thin as the sockets allow (10 mm), unless the holder would then be lower than `hook_depth` (40 mm). Up to 6 rows the hook sets the height; beyond that the slope does. `moment_self` needs the holder's mass, an estimate: side profile area × width × density × fill_factor. At 0.17 Nm for 4 × 4, the self-weight moment is far below the 1 Nm limit. Pushing a bit in at the outer row would load the cleat ten times more (1.8 Nm at 20 N); the use instructions forbid it, and Appendix D argues the case once instead of checking it in the model.
- The full calculation is in Appendix D.

4.5.7 Practical rule

A useful rule of thumb is:

> If you cannot explain what an expression does in one sentence, it is probably too complex. Break it down, name the parts and move logic to the appropriate layer.

4.6 Feature-tree and Body strategy

The organisation of Bodies, features, reference geometry and operations in the model tree affects robustness, recompute performance, maintainability and handover. The tree should make the model's dependency structure understandable: parameters and master references define the conditions, Bodies implement geometry, and a controlled result identifies the configuration intended for review, export and release.

Use a logical and consistent order:

- Control and data objects: "UIParams", "CMParams", material and process profiles, validation data and release information
- Master references: principal datums, axes, interface domain, master sketches and other stable architectural geometry
- Geometry Bodies: the stable core and, where necessary, alternative topology Bodies
- Optional or local features: details that do not alter the primary geometric strategy
- Active result and release objects: the explicitly identified result used for inspection, validation, export and release

Name important objects according to their function rather than their creation order. Names such as `Datum_BuildPlane`, `Sketch_Interface`, `Base_Shell`, `Mounting_Holes`, `Rib_Pattern` and `Body_TaperedCylinder` reveal the design logic more clearly than default names such as `Pad001` or `Sketch007`.

A clean feature tree makes it easier to:

4 The finger-access rule is applied here to orthogonal neighbours, which is conservative. In a square grid the diagonal neighbours are √2 × pitch apart, so the diagonal gap between two sockets is about 12 mm instead of 6.5 mm. Where space is critical, the creator may apply bit_spacing to the diagonal instead: `pitch = max((socket_ac + bit_spacing) / √2; socket_ac + min_wall)`, which reduces the pitch from 14.1 to about 10 mm. That is an engineering decision, not a free improvement: it assumes users grip bits diagonally, which a short grip test with real bits should confirm before the rule is changed. The point of writing the rule as an expression is that this assumption is visible and can be revised.

- Understand the build sequence and dependency direction
- Locate and modify a specific feature without disturbing unrelated geometry
- Diagnose failures when a parameter combination does not recompute
- Distinguish user configuration from protected implementation
- Hand the model over to another engineer or transfer it to an external configuration system

Avoid unnecessary intermediate Bodies, duplicate geometry, experimental features and temporary construction objects in the released model. Development aids may be retained where they have a documented engineering purpose, but they should be isolated clearly from released geometry and should not appear as part of the normal user configuration path. The basic dependency direction should remain one-way:

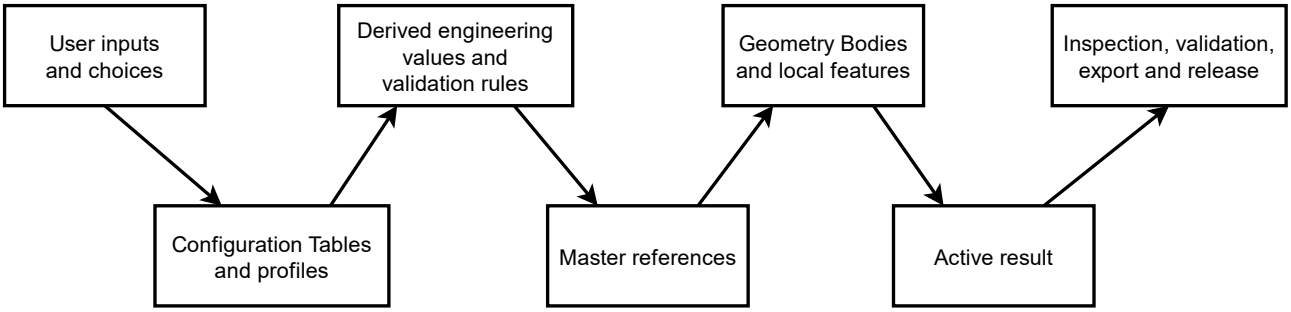


Figure 4–3: One-way dependency direction

A downstream feature should not become an unexplained source of truth for an upstream sketch, parameter or master reference. Circular dependencies obscure design intent and make the model difficult to modify safely.

4.6.1 Optional features and configuration-dependent geometry

Optional features and configuration-dependent geometry shall be isolated from the stable core of the model. A feature that is absent from one supported configuration shall not become a required reference for common downstream features.

Use optional features within a Body only when the main topology and reference strategy remain stable. Suitable examples include:

- Drainage or vent holes
- Embossed or debossed text
- Local ribs or gussets
- Chamfers and fillets
- Minor mounting details
- Locating marks, tabs or small interface features
- Cosmetic patterns that do not alter the structural or manufacturing strategy

These features should be placed after the stable core geometry, reference stable datums or local sketches, and fail independently where possible. For example, a rib pattern should not be the only reference used to position a critical mounting hole. If the rib pattern is removed, suppressed or altered, the mounting geometry should remain valid.

A configuration-dependent feature should have a clear rule for its presence, size, location and permitted range. The controlling parameter should be named and documented in the control layer rather than being represented by an undocumented manual edit in the model tree.

© 2026 Magimo

Optional geometry can be represented by a Boolean choice, a controlled discrete option or a bounded numerical rule. For example:

```
UIParams.includeDrainHoles

UIParams.edgeTreatment

UIParams.ribOption
```

The feature tree should make the effect of the option clear:

```
Body_Main

├── Base_Shell

├── Mounting_Interface

├── Drain_Holes_Optional

├── Rib_Pattern_Optional

└── Edge_Finish
```

An optional feature shall not be used to conceal a change in the primary geometric strategy. If a configuration requires a different principal sketch, base-contact geometry, print orientation, support strategy, load path, feature sequence or reference structure, it is not merely an optional feature. It is a different topology and should be implemented as a separate Body.

> **Bit holder**
>
> One socket is modelled at the first position. One linear pattern copies it in two directions: outwards (Occurrences = rows_out) and across (Occurrences2 = rows_across), both at pitch. The model never needs a list of positions: the pattern does the counting.
>
> The holder row is a built-in feature, not optional: a second socket sketch with its own deeper pocket (`qh_depth`) and a one-direction pattern across.

4.6.2 Alternative topology Bodies

Use separate Bodies for geometry variants that require different primary sketches, feature sequences, reference structures, base-contact geometry, print orientation, support strategy or load path. Each Body should represent one defined topology and should consume shared values from the central control layer. A suitable project structure is:

```
Project
│
├── UIParams
├── CMParams
├── MaterialProfile
├── ProcessAspects
├── Validation
├── ReleaseInfo
│
├── MasterReferences
│   ├── Datum_BuildPlane
│   ├── Axis_Centre
│   ├── Plane_Interface
│   └── Sketch_InterfaceGeometry
│
├── Geometry_Variants
│   ├── Body_Cube
│   ├── Body_TaperedCuboid
│   └── Body_TaperedCylinder
│
└── ActiveResult
```

Alternative topology Bodies should not depend on one another unless the dependency is deliberate, stable and part of the defined architecture. A Body should reference only:

- Central Spreadsheet aliases and controlled parameters
- Material, process and validation data
- Stable datum geometry and master-reference geometry
- Its own local sketches, features and construction geometry

Do not reference generated faces, edges or vertices from one topology Body in another. Such references create cross-Body coupling and expose the model to failures when one variant is changed, suppressed, reordered or removed. The architectural principle is:

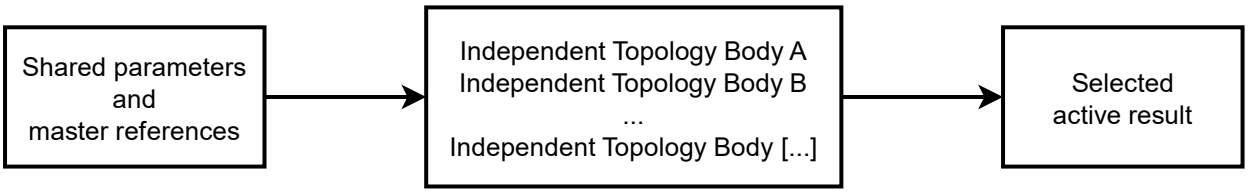


Figure 4–4: Architectural principle of alternative Bodies

Alternative Bodies may share the same interface geometry, mounting locations, overall dimensions, material profile and process assumptions, but they are free to implement the geometry differently. This allows each topology to remain understandable and robust within its own modelling strategy. For example, a cube, a tapered cuboid and a tapered cylinder may all share an external interface and a common parameter set, while requiring different sketches, lofts, pad features, wall transitions or support strategies. Forcing them into one Body through extensive suppression logic can create a fragile feature history in which every variant depends on features that are irrelevant or invalid for the others.

Each topology Body should be tested independently across the applicable parameter range. A configuration is not supported merely because one alternative Body has been validated. The evidence must apply to the selected topology, material profile, process profile, orientation and intended use.

4.6.3 Visibility Toggle as a configuration selector

A Visibility Toggle is a high-level configuration selector. It determines which approved geometry variant is the active result without requiring the user to understand, edit or navigate the separate Bodies that implement those variants.

Visibility does not suppress computation. Making a Body, feature or object invisible changes only its presentation in the current view. It does not by itself remove the object from the model's dependency graph, prevent its recompute or guarantee that it is excluded from validation, Boolean operations, bounding-box calculations, compounds or export workflows. A hidden object may therefore still consume calculation time, contribute to errors, participate in dependencies or influence an exported result, depending on the object type and workflow.

> Visibility answers "what is shown"; configuration control answers "what is active, checked, exported and released". The Visibility Toggle does not establish that a configuration is valid. The active Body must still recompute correctly and pass the applicable geometry, manufacturing and supported-domain checks. Export and release shall use the explicitly identified active result rather than relying only on the graphical visibility state.

4.7 Scaling versus genuine parametric behaviour

Global scaling alone is not sufficient to constitute genuine parametric behaviour. Scaling does not adjust:

- wall thickness relative to nozzle and process;
- clearances and fits;
- rib spacing and feature density;
- print orientation and support strategy; or
- structural proportions.

In this section, global scaling means applying one scale factor to the entire model, normally through the slicer or another downstream manufacturing tool. Every dimension is multiplied by the same factor, including functional dimensions, wall thicknesses, clearances, feature sizes and tolerances. Global scaling therefore changes the complete geometry without applying separate engineering rules to the dimensions that may need to remain fixed or change differently. A model that relies on global scaling to achieve different sizes is therefore not truly parametric in the sense used in this guide. Genuine parametric behaviour means:

- key dimensions change according to meaningful inputs (e.g. hand width, mating diameter, installation space);
- derived dimensions (wall thickness, clearances, rib spacing) adjust according to engineering rules; and
- the model remains manufacturable and functional across the supported range.

Global scaling can be useful when the design is geometrically self-similar and its functional, manufacturing and material assumptions remain valid at the new size. It is not a substitute for parametrisation when dimensions have different roles or must obey different rules.

> **Bit holder**
>
> Scaling the holder by 120 % to “fit more bits” breaks two interfaces at once. The sockets grow from 6.6 mm to about 7.9 mm across flats, so the bits fall out, and the hook no longer matches the 45° cleat on the wall. More bits means more rows, never a larger scale. The release notes should mention this.

4.8 Performance and maintainability

A robust parametric model should be efficient as well as correct. Performance and maintainability affect whether a model can be practically configured, tested, reviewed and handed over. A well-structured model should:

- avoid unnecessarily complex sketches or feature trees that slow recomputation;
- simplify geometry where detail does not contribute to function, manufacturability or required product information;
- use patterns and arrays judiciously, because high instance counts can significantly increase recompute and export time;
- separate user inputs, engineering logic, derived results and validation or release information;
- remove or clearly isolate debugging geometry, test sketches, experimental features and diagnostic objects from released models.

Performance is not only a technical convenience. A model that takes minutes to recompute, produces unclear dependencies or cannot be understood by another engineer is less likely to be used, tested and maintained as intended. Maintainability should therefore be treated as part of the parameter architecture. The objective is not the smallest possible model, but a model whose structure remains understandable and reliable as configurations, variants and contributors change.

© 2026 Magimo

5 Encode Manufacturing in the Model

This chapter builds the printing process into the model instead of leaving it to the slicer. It shows how material, orientation, walls, overhangs, fits and tolerances become parameters and rules, and how manufacturing profiles keep those assumptions together.

5.1 DfAM is a design input, not a slicer correction

Design for additive manufacturing (DfAM) must be treated as a design input, not as a post-processing step in the slicer. If manufacturing constraints are only considered after the geometry is finalised, the model will either:

- require slicer workarounds that compromise performance or reliability; or
- generate configurations that are mathematically valid but physically unsound.

In a parametric DfAM model, manufacturing knowledge should be encoded in:

- the parameter architecture (e.g. minimum wall rules, clearance formulas);
- the geometry (e.g. self-supporting angles, rib anchoring);
- the expressions (e.g. nozzle-dependent offsets, material-specific compensations); and
- the validation logic (e.g. checks for unsupported spans, overhangs or heat accumulation).

The slicer remains an important part of manufacturing, but it should not be expected to correct fundamental design errors that belong in the model.

5.2 Process context as parameter input

Manufacturing constraints depend on the process context. A robust model therefore treats process information as explicit parameter input, not as implicit assumptions. Key process parameters typically include:

- printer type and capabilities (e.g. FDM, SLS, MJF);
- nozzle diameter (for extrusion-based processes);
- layer height;
- material family and grade;
- intended print orientation;
- support strategy (none, limited, permitted);
- infill type and density (if relevant to stiffness or weight); and
- post-processing requirements (e.g. support removal, heat treatment).

Some of these values may be:

- supplied by the user (e.g. selected material, available printer);
- fixed by the creator (e.g. required orientation for load-critical parts); or
- selected from a controlled manufacturing profile (e.g. a defined combination of printer, nozzle, material and process settings).

The model should not assume a “standard” 0.4 mm nozzle, 0.2 mm layer height or PLA material unless these are explicitly stated and justified. Relevant process parameters should be

© 2026 Magimo

represented in the interface, stored in a controlled manufacturing profile or documented as fixed assumptions.

Bit holder

The creator's print basis fixes `nozzle_d` (0.4 mm), `line_w` (0.45 mm) and a 0.2 mm layer height.The build volume comes from the user; the creator's `bed_margin` (e.g. 5 mm per side, for skirt, brim and clips) turns it into the usable envelope that bounds the design. Which axis the plate is checked against depends on the print orientation.

The profile also drives the engineering rules. A printer builds a wall from side-by-side lines of extruded plastic, each one line_width wide. A wall that is not a whole number of lines leaves a narrow gap that the slicer fills poorly or not at all, so `min_wall` is set to a whole number of lines: 4 × 0.45 = 1.8 mm, two lines from each socket.

The same logic sets the socket floor to a whole number of layers, at least 6; the model uses `floor_t = 2 mm` (10 layers). The nozzle is not a user input: the rules and tests hold only for this print basis, so the slicer settings must match it. `min_wall = 4 × line_w` follows automatically if the creator changes the basis (see §5.5.1 and §E.3.3.3).

5.3 Material and process variation

Material and process variation are central to DfAM because the same nominal geometry can behave differently depending on the selected material and manufacturing conditions. Where these differences affect function, fit, strength, durability or service conditions, they should be reflected in the design basis or documented as limitations. Different materials and processes behave differently. A parametric DfAM model should account for at least the sources of variation described in the following sections.

5.3.1 Shrinkage and dimensional change

If shrinkage materially affects the intended application, the model should include or document an appropriate compensation approach. The compensation value should be associated with the relevant material, printer, process profile and measurement basis. A generic factor should not be presented as universally applicable. For tight fits or precision interfaces, shrinkage compensation may be critical. For non-critical cosmetic parts, it may be acceptable to ignore it – but this should be a deliberate decision, not an oversight.

5.3.2 Moisture sensitivity

Some materials (e.g. PA/Nylon, certain composites) are moisture-sensitive. Moisture affects:

- dimensional stability;
- mechanical properties;
- surface quality; and
- long-term performance.

If the design is intended for moisture-sensitive materials, the creator should:

- specify drying and storage requirements in the handover information;
- consider the effect of moisture absorption on critical dimensions and fits; and
- where necessary, design for worst-case moisture conditions or restrict the material selection accordingly.

5.3.3 Layer bonding and mechanical behaviour

In layer-based processes, mechanical properties are not isotropic. In many extrusion-based polymer prints, strength within deposited layers differs from strength across layer interfaces. The size and direction of that difference depend on the material, process settings, geometry and test method. The model should reflect these realities by:

- orienting load-critical features so that principal loads lie in the stronger XY plane where possible;
- avoiding designs that rely on Z-direction strength unless explicitly validated;
- using ribs, gussets and transitions to distribute loads; and
- documenting the assumed load directions and orientation requirements.

5.3.4 Stiffness, creep and temperature

Polymers exhibit:

- viscoelastic behaviour (creep under sustained load);
- temperature-dependent stiffness and strength;
- potential softening near the glass transition temperature; and
- long-term property changes due to ageing or environmental exposure.

For functional parts, the creator should:

- consider the expected load duration and temperature range;
- avoid designs that rely on marginal stiffness or strength;
- where necessary, include safety factors that account for creep and temperature effects; and
- document the intended service conditions and restrictions of use.

5.3.5 Material profiles and configuration control

A practical way to manage material and process variation is through material profiles. Each profile defines:

- material family and grade;
- supported nozzle and layer-height ranges;
- shrinkage or compensation assumptions;
- mechanical and thermal property classifications;
- chemical and environmental restrictions; and
- validation status (e.g. tested, supplier data, engineering judgement).

The user may select a material profile from a permitted list. The model then retrieves the associated properties and applies the relevant rules. The user should not edit the underlying material properties unless explicitly allowed as part of an advanced configuration.

> **Bit holder**
>
> The holder hangs under constant load in a garage that can be warm in summer. That makes creep the material question, not strength.
>
> - Restricted: PETG only. ASA is an alternative candidate once it has its own coupon and creep test.

- Prohibited: PLA. It creeps under constant load and softens well below the temperatures a garage can reach.
- Not indifferent: material also changes how much a hex socket shrinks, so `socket_clearance` depends on the material. With a second material, a material table becomes a natural Configuration Table (see Appendix C); with one material, density is a constant in CMParams (density of PETG = 1.27 g/cm³).

The Instructions for Manufacturing should stipulate that PETG must be dried before printing: moist filament weakens the layer bond exactly where the hook is loaded.

5.4 Orientation, anisotropy and load path

Print orientation is one of the most important manufacturing decisions in FDM and similar processes. It affects:

- mechanical anisotropy (strength and stiffness in different directions);
- surface finish on different faces;
- support requirements and removal effort;
- dimensional accuracy (especially in Z); and
- the ability to print certain features without defects.

5.4.1 Orientation and anisotropic strength

In extrusion-based additive manufacturing, orientation affects how material is deposited relative to the expected loads. It can influence strength, stiffness, failure mode, surface quality, dimensional behaviour, support requirements and post-processing. The relevant relationship must therefore be considered during design rather than left entirely to the slicer. This means:

- tensile and bending loads should preferably be oriented in the XY plane;
- Z-direction loads should be minimised or explicitly validated;
- features such as holes, bosses and ribs should be designed with this anisotropy in mind; and
- the required orientation should be stated clearly in the handover information.

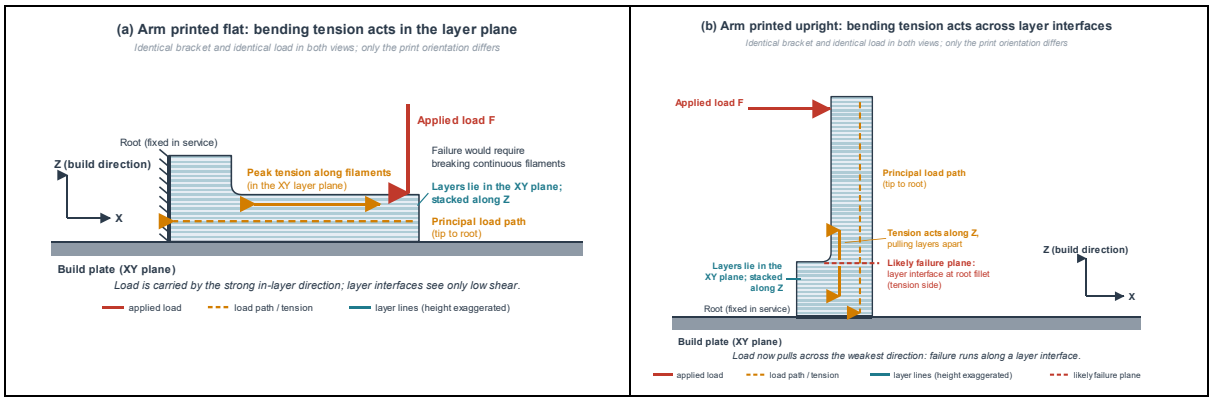


Figure 5–1: Effect of print orientation on load path in an FDM bracket

The same geometry can place its principal bending load mainly within the layer plane or across layer interfaces, depending on orientation. The suitable orientation must be determined for the

material, process and loading conditions and validated where the part is load-critical. If the design is load-critical, the creator should:

- define the principal load direction;
- specify the required print orientation to align loads favourably;
- where necessary, restrict user choice of orientation or flag non-standard orientations as unsupported; and
- document the evidence (e.g. test results, supplier data, engineering analysis) that supports the orientation requirement.

5.4.2 Surface finish and dimensional behaviour

Different faces of a part will have different surface qualities depending on orientation:

- surface quality depends strongly on whether a surface is formed by side walls, upward-facing layers, downward-facing layers, bridges or supported regions;
- downward-facing surfaces may require supports and can show support marks;
- side walls show layer lines and may have stair-stepping on sloped surfaces; and
- critical mating or bearing surfaces may need post-processing.

The model should:

- identify critical surfaces in the documentation;
- where possible, orient the part so that critical surfaces are upward-facing or otherwise favourable;
- account for any additional material or clearance needed for post-processing; and
- communicate these requirements to the user and manufacturer.

5.4.3 Supports and orientation-dependent geometry

Supports affect:

- print time and material usage;
- surface quality on supported faces;
- the risk of damage during support removal; and
- the feasibility of certain geometries (e.g. steep overhangs, bridges).

The model should:

- avoid features that require excessive or difficult-to-remove supports where possible;
- design self-supporting angles where possible. A commonly used starting point is around 45° from the build direction, but the acceptable limit depends on the printer, material, line width, layer height, cooling, speed, geometry and required surface quality. Treat the value as a process-specific starting point rather than a universal rule;
- where supports are unavoidable, identify the supported faces and provide guidance on removal; and
- where the design depends on a specific support strategy (e.g. no supports on a mating face), make this a stated requirement.

© 2026 Magimo

5.4.4 Orientation as part of the release contract

For many parametric products, orientation is not an optional preference – it is part of the release contract. The creator may:

- require a specific orientation (e.g. “Face A on build plate”);
- permit a small set of validated orientations; or
- declare that orientation is critical and must not be changed without re-validation.

This requirement should be:

- stated in the model documentation;
- reflected in any manufacturing manifest or handover file; and
- enforced through warnings or configuration restrictions where appropriate.

Bit holder: one orientation, released with the model

The holder is printed with its top face on the bed. The sockets then print vertically, as accurate hexes, and each socket floor bridges about 7 mm without support. The underside of the hook is a 45° overhang, which also prints without support.

The other orientations each cost something. On its back or on its side, the sockets print horizontally: the hexes come out less accurate, and the fit of the bits varies. Some of these orientations also need support. Printing the hook as a separate part adds a joint, an interface and an assembly step.

In the chosen orientation the hook bends across the layers, its weakest direction. The stresses are low (Appendix D), so this is acceptable. The orientation is part of the release contract. It fixes socket_clearance, which is valid for vertical holes only, and it sets which axis the envelope check uses for the height.

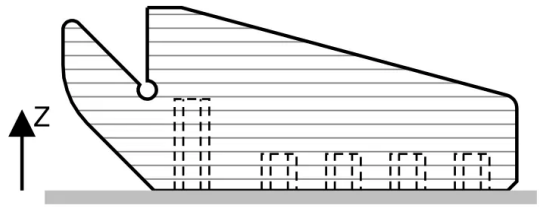


Figure 5–2: The bit holder as printed, top face on the bed. The sockets (dashed) open onto the bed and print vertically.

5.5 Walls, features, bridges and support strategy

Walls, ribs, bosses, bridges and overhangs are the building blocks of many FDM parts. Their design must reflect the capabilities and limits of the process.

5.5.1 Walls and shells

Key considerations for walls and shells include:

- minimum wall thickness as a function of nozzle diameter and material;
- the relationship between thickness and the process steps (walls as whole multiples of the line width; floors and ledges as whole multiples of the layer height);
- the effect of wall thickness on stiffness, weight and print time;

- transitions between different wall thicknesses (use fillets or tapers to avoid stress concentrations and print defects); and
- the risk of warpage or heat accumulation in thick or densely packed walls.

A robust model will:

- enforce minimum wall rules based on the selected process and nozzle;
- treat wall thicknesses as a creator-controlled rule or constrained design value unless the user has a genuine, supported reason to specify it directly.
- use transitions and fillets where wall thickness changes; and
- flag or restrict configurations that violate manufacturing limits.

5.5.2 Minimum features

Minimum feature sizes depend on the process and nozzle. Typical considerations include:

- minimum hole diameter (affected by shrinkage, support and nozzle size);
- minimum pin or boss diameter (affected by strength and printability);
- minimum gap or clearance (affected by nozzle, layer height and material);
- minimum rib thickness and spacing (affected by heat accumulation and warpage); and
- minimum text size and depth (if embossed or debossed text is used).

The model should:

- define minimum feature rules based on the supported process and nozzle;
- apply these rules consistently across all features;
- where necessary, adjust feature sizes automatically to stay within manufacturable limits; and
- document the assumptions and any known exceptions.

5.5.3 Bridges and overhangs

Bridges and overhangs are limited by:

- the maximum unsupported span for the given nozzle, material and layer height;
- the angle of the overhang relative to the build plane;
- the availability of underlying geometry to support the feature; and
- the desired surface quality (bridges and overhangs often show sagging or roughness).

The model should:

- limit bridge spans to values that are known to be printable under the stated conditions;
- keep bridges or overhangs, where unavoidable, within validated limits; and
- document any orientation or support requirements associated with these features.

5.5.4 Support strategy

Supports are sometimes necessary, but they come with trade-offs:

- increased print time and material usage;

© 2026 Magimo

- potential damage or marks on supported surfaces;
- additional post-processing effort; and
- risk of incomplete support removal in complex geometries.

The model should:

- minimise the need for supports through geometry and orientation choices;
- where supports are required, design the part so that supports are accessible and removable;
- avoid placing supports on critical mating or bearing surfaces where possible;
- state the expected support strategy in the handover information; and
- where the design assumes a specific support approach (e.g. tree supports, no supports on certain faces), make this explicit.

5.5.5 Rule hierarchy

Manufacturing rules should follow a clear hierarchy:

- Hard rules – non-negotiable limits that must be enforced (e.g. minimum wall thickness, maximum overhang angle). Violating these should result in a configuration rejection or clear failure.
- Soft guidance – recommendations that improve printability or performance but are not strictly required (e.g. preferred infill density, suggested orientation for cosmetic quality). Violating these may be acceptable with user awareness.
- Human review – cases where the model cannot fully automate the decision and a human check is required (e.g. unusual material combinations, borderline configurations).

The model should distinguish between these categories and respond appropriately: enforce hard rules, communicate soft guidance and flag cases that require human review.

> **Bit holder**
>
> - Walls between sockets: `min_wall = 1.8 mm`.
> - Socket floors: printed with the plate top on the bed, each floor bridges about 7 mm. That prints cleanly without support, and any sag stays hidden inside the socket. The mouths get a 0.5 mm chamfer against the first-layer elephant foot. [5]
> - Hook root: a corner relief (dogbone) of r = 2 mm at the 45° root. A fillet would clash with the sharp edge of the wall cleat; the relief gives the edge room and lowers the stress. A sharp root multiplies the stress there by roughly 2–3.
> - Supports: none. In orientation A the underside of the hook is a 45° overhang, which prints without support.

5.6 Fits, clearances, interfaces and tolerance stack-up

Fits and clearances are critical for assembly, function and safety. In additive manufacturing, they are affected by:

5 Elephant foot (also called elephant's foot): the first layers of a print spread slightly outwards, because the nozzle presses them onto the bed and the heated bed keeps them soft. Outer edges grow and openings shrink, typically by 0.1–0.3 mm. A small chamfer on edges at the bed compensates for the effect.

- process capability (nozzle size, layer height, machine accuracy);
- material behaviour (shrinkage, warpage, moisture);
- orientation (dimensional accuracy in Z vs XY);
- feature size and geometry; and
- post-processing (e.g. reaming holes, sanding surfaces).

5.6.1 Define the interface first

Before specifying clearances, define the interface:

- what is mating with what (e.g. pin–hole, slot–tab, threaded insert);
- the required type of fit (e.g. loose, normal, tight, press);
- the load and movement conditions (e.g. static, sliding, rotating);
- the environmental conditions (e.g. temperature, humidity, chemical exposure); and
- the assembly method (e.g. hand assembly, tool-assisted, adhesive).

The clearance should then be derived from:

- the nominal size of the mating features;
- the selected fit class;
- the process and material assumptions; and
- any additional allowances for post-processing or tolerance stack-up.

5.6.2 Make clearances process-aware

Clearances that work for one process or material may not work for another. A clearance value may be suitable for one combination of nozzle, material, printer and process profile but unsuitable for another. Any numerical example should therefore be identified as illustrative unless it is supported by evidence for the stated conditions. Examples of process-dependent effects include:

- A clearance such as 0.2 mm may be suitable for a particular 0.4 mm-nozzle and PLA process profile, but this is illustrative only and is not a universal clearance value.
- Dimensions in the Z direction may have different tolerances from dimensions in the XY plane.
- Holes may print smaller than nominal because of process behaviour or shrinkage and may require compensation.
- External features may print larger or smaller than nominal depending on extrusion width, flow, cooling and other process conditions.

The model should:

- use clearance rules that account for the relevant nozzle, material, printer, process profile and orientation;
- differentiate between XY and Z clearances where necessary;
- allow clearance rules to vary with the material or process profile; and
- document the assumptions behind the clearance values.

© 2026 Magimo

5.6.3 Keep interfaces consistent

Where a part has multiple interfaces (e.g. several mounting holes, multiple mating features), keep the clearance and tolerance approach consistent unless there is a specific reason to do otherwise. Inconsistent clearances can:

- confuse the user or manufacturer;
- lead to assembly problems; and
- make the model harder to validate and maintain.

If different interfaces require different fits, this should be:

- explicitly defined in the model;
- clearly documented; and
- reflected in the parameter structure (e.g. separate fit classes for different interfaces).

5.6.4 Tolerance stack-up

In assemblies with multiple parts or features, tolerance stack-up can become significant. Consider:

- the cumulative effect of multiple clearances and tolerances;
- the direction of stack-up (e.g. along a critical axis);
- the worst-case combination of tolerances; and
- whether the design can tolerate the expected variation.

Where stack-up is critical, the model should:

- use conservative clearances or adjust feature sizes accordingly;
- where possible, design for adjustability or compliance to absorb variation; and
- document the expected stack-up and any assembly instructions needed to manage it.

5.6.5 Interfaces beyond holes and pins

Interfaces are not limited to holes and pins. They may include:

- snap-fits and living hinges;
- threaded inserts and heat-set inserts;
- adhesive bonds and welded joints;
- friction fits and interference fits; and
- custom mating geometries (e.g. dovetails, keyways).

Each interface type has its own design rules and manufacturing considerations. The model should:

- encode the relevant rules for each interface type;
- ensure that the geometry remains within printable and functional limits; and
- document any special instructions (e.g. insert installation torque, adhesive curing conditions).

Bit holder: three interfaces, three kinds of fit

1. Bit in socket: hex holes print undersize. Starting values on the across-flats size, vertical sockets in PETG: e.g. +0.25 mm for a normal fit. The coupon replaces it. (See §6.3.6).
2. Hook on cleat: the rear face sits 18.5 mm from the wall, 0.5 mm more than the nominal cleat. A wooden cleat varies, e.g. ±0.5 mm in thickness and ±1–2° in bevel.The hook must still bear at the top and the back must still touch the wall at the bottom; if the hook is too deep, the holder tilts forwards. That is a small tolerance stack (§5.6.4).
3. Lift-off: to lift the holder off, it must rise by the hook engagement plus a small clearance. That needs free space above the holder on the wall, which the use instructions should state.

5.6.6 Scaling and tolerance behaviour

Global scaling applies one factor to every geometric dimension. It therefore scales functional dimensions, wall thicknesses, clearances, feature sizes and nominal tolerance dimensions together. This may be acceptable for a geometrically self-similar design, but it can invalidate fits, clearances, process assumptions or tolerance relationships when those dimensions do not have the same scaling requirement. If size variation is needed, it should be implemented through:

- controlled parameters that adjust nominal dimensions while keeping clearances and tolerance requirements process-aware; or
- separate parameter-driven rules for different feature types, such as scaling the overall size while deriving clearances from the nozzle, material, orientation and process profile.

Avoid using simple uniform scaling of the final geometry as a general method for producing different sizes.

5.6.7 Release requirement

For released parametric products, the fit and clearance strategy should be:

- defined in the model (through parameters, expressions and geometry);
- documented in the handover information (e.g. intended fit classes, assumed process capability);
- validated through print trials and assembly tests for representative configurations; and
- communicated to the user and manufacturer as part of the release contract.

5.7 Hard rules, soft guidance and exceptions

Not all manufacturing constraints are equally critical. A useful distinction is between hard rules, soft guidance and exceptions.

5.7.1 Hard rules

Hard rules are non-negotiable limits that must be enforced to ensure printability, function or safety. Examples include:

- minimum wall thickness for the selected nozzle and material;
- maximum overhang angle without supports;
- minimum clearance for a given fit class and process;
- maximum unsupported bridge span; and
- orientation requirements for load-critical parts.

© 2026 Magimo

Violating a hard rule should result in:

- a configuration rejection (the model does not build or explicitly flags the error); or
- a clear warning that the configuration is outside the supported domain and should not be used.

Hard rules should be:

- encoded in the model where possible (e.g. through expressions, constraints or validation checks);
- documented in the handover information; and
- validated through testing or supported by supplier data and engineering analysis.

5.7.2 Soft guidance

Soft guidance consists of recommendations that improve printability, performance or aesthetics but are not strictly required. Examples include:

- preferred infill density and pattern;
- suggested orientation for cosmetic quality;
- recommended layer height range for surface finish; and
- advice on support density and removal.

Violating soft guidance may be acceptable if the user understands the trade-offs. The model should:

- communicate soft guidance clearly in the documentation;
- where appropriate, include it in the manufacturing manifest; and
- avoid presenting it as a hard requirement unless there is a specific reason.

5.7.3 Human review

Some cases cannot be fully automated and require human judgement. Examples include:

- unusual material combinations;
- borderline configurations that are technically printable but not well validated;
- applications with special safety or regulatory requirements; and
- custom modifications outside the standard configuration domain.

The model should:

- flag cases that require human review;
- provide enough information for a competent person to make an informed decision; and
- document the outcome of the review where relevant.

> **Bit holder**
>
> - Hard rules (reject): the holder fits the usable envelope in the released orientation; the moment of the filled holder stays within allowable_moment. Built into the rules: `min_wall` ≥ 4 lines; no supports needed.

- Soft guidance (warn): none in the running example. `bit_spacing` is a creator rule inside pitch, so a pitch too narrow for fingers cannot occur.
- Human review: `rows_out` above 6 until the creep test has passed (§2.3, §6.3.5).

The difference matters to the user: a hard rule stops the configuration, soft guidance explains a trade-off and leaves the choice open.

5.7.4 Exceptions must be explicit

If an exception to a hard rule is deliberately allowed (e.g. a thinner wall for a non-critical cosmetic feature), this must be:

- explicitly defined in the model;
- clearly documented;
- limited to the specific case and conditions for which it is justified; and
- distinguished from the general rule.

Exceptions should not become a backdoor for bypassing important constraints. They should be rare, well-justified and tightly controlled.

5.7.5 Rule precedence

When rules interact or conflict, there should be a clear precedence. For example:

- hard rules take precedence over soft guidance;
- safety-related rules take precedence over cosmetic or convenience considerations;
- validated configurations take precedence over untested ones; and
- documented exceptions take precedence over general rules for the specific case.

The model and documentation should make this precedence clear so that users and manufacturers know which rules to prioritise.

5.8 Manufacturing-aware profiles

A manufacturing profile may have its own validation and release status. A supported geometry variant is not automatically supported for every material or process profile. The applicable status shall be determined for the combination of topology, parameter range, material profile, process profile, orientation and intended use.

A practical way to manage manufacturing complexity is through manufacturing-aware profiles. Each profile defines a coherent set of process assumptions and constraints.

5.8.1 What a profile contains

A manufacturing profile should use the same status vocabulary as the overall model where practical or clearly define any profile-specific status terms. At minimum, distinguish between a profile that is available for development, a profile that has supporting evidence, a profile that is supported for release and a profile that is deprecated or no longer authorised for new use. Such a profile may include:

- printer type and model (or class of printers);
- nozzle diameter and type;
- layer height range;

- material family and grade (linked to material profiles);
- orientation requirements;
- support strategy (e.g. none, limited, permitted; specific support types);
- infill type and density (if relevant);
- process settings (e.g. temperatures, speeds – if they affect geometry or validation);
- dimensional compensation rules (e.g. hole oversize, shrinkage factors);
- validation status (e.g. tested configurations, representative test cases); and
- limitations and known issues.

Profiles allow the creator to:

- package complex assumptions into manageable units;
- offer users a choice between validated manufacturing options; and
- keep the base model intact while varying the manufacturing context.

5.8.2 Keep the base model intact

The base parametric model should remain independent of any single profile. Profiles should:

- be applied as a layer on top of the base model;
- adjust parameters, rules and validation checks without changing the core geometry logic; and
- be selectable or switchable without requiring a complete redesign.

This allows the same base model to support multiple printers, materials and process combinations through different profiles.

5.8.3 Profiles contain coupled assumptions

Profiles are not just a list of independent settings. They represent coupled assumptions about how the part will be made and used. For example:

- a profile for a high-temperature material may assume a different orientation and support strategy than a profile for a standard material;
- a profile for a fine-detail printer may allow smaller features than a profile for a production printer; and
- a profile for a specific material–printer combination may have been validated for certain configurations but not others.

When creating or modifying a profile, consider the interactions between:

- material behaviour;
- printer capabilities;
- process settings;
- geometry rules; and
- validation evidence.

5.8.4 Profile status

Each profile should have a clear status, such as:

- Draft: under development and not released for general use;
- Experimental: available for investigation but not generally supported;
- Validated: supported by identified evidence for stated conditions;
- Supported: approved for the declared model and process domain;
- Released: included in an authorised release combination;
- Deprecated: still identifiable but not recommended for new configurations;
- Obsolete: no longer maintained or authorised for new use; and
- Rejected: failed a requirement or is explicitly prohibited.

The status should be:

- recorded in the model or accompanying documentation;
- visible to the user where relevant; and
- updated when the profile is re-validated or retired.

5.8.5 Profile selection and output

The user or manufacturer selects a profile based on:

- available equipment;
- material stock;
- required quality and performance; and
- any customer or regulatory requirements.

The model should:

- use the selected profile to adjust parameters, rules and validation checks;
- generate a manufacturing manifest or handover file that includes the profile identifier and key settings; and
- where appropriate, warn if the selected profile is not compatible with the chosen configuration.

5.8.6 Embedded and external profiles

Profiles can be:

- Embedded – stored within the FreeCAD file (e.g. in a dedicated Spreadsheet or data structure); or
- External – stored in separate files, databases or platform services and referenced by the model.

Embedded profiles are convenient for self-contained models. External profiles are useful when:

- managing many profiles across multiple models;
- integrating with a print service or platform; or
- updating profiles independently of the model.

© 2026 Magimo

External profiles should be identified by stable names or identifiers, versioned and included or referenced in a way that allows the released configuration to be reconstructed.

The choice depends on the use case. The important point is that the profile mechanism is clear, documented and consistent.

5.8.7 Do not overstate profile coverage

A profile does not automatically validate every possible configuration. The creator should:

- state which configurations have been tested with each profile;
- identify any known limitations or unsupported combinations;
- avoid implying that a profile makes the model suitable for all materials, printers or applications; and
- update the profile coverage as additional testing and evidence become available.

Profiles are a tool for managing complexity, not a substitute for engineering judgement and validation.

6 Verify, Control and Release

This chapter turns a working model into a released product. It defines what Validated, Supported and Released mean, how to gather evidence through boundary tests and print trials, and how to control changes, document the model and package it for release.

6.1 What “validated” means for a parametric product

For a conventional CAD model, validation may be understood as confirming that the geometry meets the design intent at one nominal configuration. For a parametric product, validation must cover a defined domain: a range of configurations, parameter combinations, materials, process profiles, orientations and intended use conditions. A successful recompute is necessary but not sufficient. It confirms only that the CAD system can generate the current model state. It does not prove that the resulting part is printable, dimensionally suitable, structurally adequate, durable in service, safe for its intended application or within the creator’s declared responsibility.

A successful recompute confirms only that the CAD system generated geometry. Geometry checks, physical or functional evidence, a defined support decision and release control are separate steps; none should be inferred from recompute success alone.

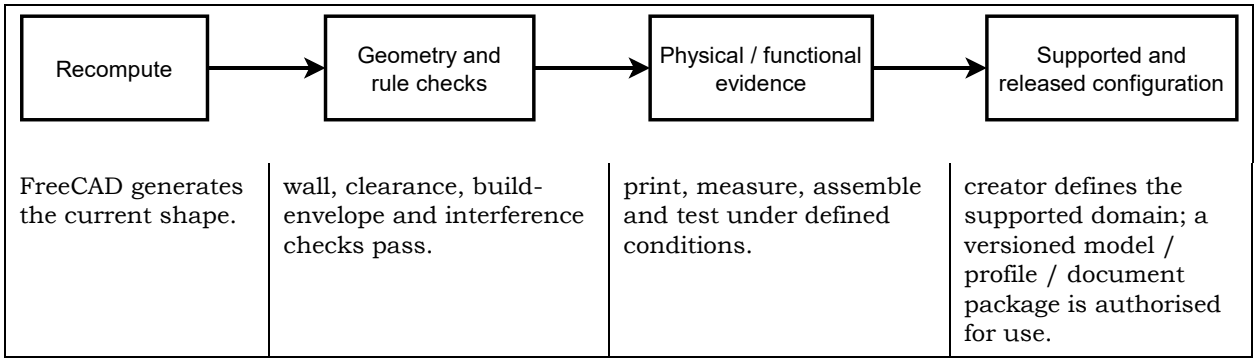


Figure 6–1: Increasing evidence and control from recompute to release

A parametric product is validated only to the extent that evidence exists for its defined conditions. Relevant evidence may include parameter-boundary tests, coupled-extreme tests, representative configurations, print trials, dimensional measurements, assembly trials, functional tests, inspection records, analysis and documented technical review. Validation is therefore not a one-time event. It is a controlled process that establishes evidence for a stated configuration domain and informs the decision to support and release that domain.

6.1.1 Status terms

Status terms distinguish between “evidence”, “declared support scope” and “release control”. They shall not be treated as interchangeable labels.

Table 6–1: Status terms

Status	Meaning	Typical basis	User-facing status	Release position
Draft	Under development and not released for general use	Working model, preliminary geometry, incomplete design decisions, open issues	No	Not for manufacture or distribution

Status	Meaning	Typical basis	User-facing status	Release position
Experimental	Available for investigation or testing, but not generally supported	Trial geometry, exploratory print, comparison study, early prototype	Normally no	Not released as a supported option
Validated	Supported by identified evidence for stated conditions and configurations	Test records show that stated requirements were met under stated conditions	Not necessarily	Evidence is available; release remains a separate decision
Supported	Included within the creator's declared configuration, manufacturing and use domain	Adequate validation evidence, documented assumptions and limitations, explicit support decision	Yes	Eligible for release, subject to release checks
Released	Included in an authorised release combination with identified versions and status.	Identified revision, controlled dependencies, correct documentation, recorded status, release approval	Yes	Current authorised output
Deprecated	Still identifiable and possibly usable for existing work, but not recommended for new configurations	Superseded design, changed requirements, preferred replacement, reduced support	Normally no for new configurations	Retained for history or existing users; not preferred for new release
Obsolete	No longer maintained or authorised for new use	Withdrawal decision, incompatible change, unacceptable limitation, replacement release	No	Not authorised for new manufacture or distribution
Rejected	Failed a requirement or is explicitly prohibited	Failure record, nonconformance, unsafe behaviour, unmanufacturable geometry, out-of-domain condition	No	Shall not be released

These terms describe different aspects of configuration control and evidence. They are not interchangeable. Validation describes the evidence available; support describes the creator's declared scope; release describes authorisation and version control.

6.1.2 Validated

"Validated" describes the evidence available for a stated configuration, manufacturing process and intended use. It does not mean that every possible configuration of the model has been validated, nor that the result is suitable for every application. A configuration is validated only for the conditions covered by the evidence. These conditions may include:

- The topology or geometry variant
- The parameter values and permitted ranges
- Important coupled parameter combinations
- The material profile
- The printer and process profile

© 2026 Magimo

- Nozzle diameter and layer height where relevant
- Required print orientation and support strategy
- Post-processing assumptions
- Intended load, environment, interface and use conditions

Validation supports specific claims under specific conditions. It does not automatically extend to different materials, printers, process settings, orientations, dimensions, loads, environments or uses. Similarly, validation of one topology Body does not validate another Body merely because both use the same control layer or interface dimensions. Furthermore, validation records shall identify what was tested, which requirements were assessed, how the test was performed, the observed result and any limitations. A configuration outside those recorded conditions shall not be described as validated merely because it recomputes or resembles a tested configuration.

6.1.3 Supported

“Supported” describes the configuration and conditions that the creator has chosen to offer and document. A supported configuration should have a defined design basis, identified manufacturing assumptions, stated limitations and an appropriate level of evidence.

A configuration may be supported even when the available evidence is not a complete qualification for every possible use. The support statement must therefore make clear what has been checked, under which conditions and what remains the responsibility of the manufacturer or user.

A configuration may be validated without being supported. For example, a topology variant may have been printed successfully for investigation, comparison or development, but remain unavailable to users because:

- The test coverage is too limited
- The manufacturing window is too narrow
- The required process is impractical for normal users
- The documentation is incomplete
- The creator does not intend to maintain the variant
- The variant is being retained only for technical comparison

A configuration shall not normally be described as supported unless adequate validation evidence exists for its intended use. “Supported” does not mean that every possible parameter combination, material or print process is acceptable. It applies only within the stated configuration domain.

6.1.4 Released

“Released” describes a controlled combination of model version, parameter schema, manufacturing profile, documentation and supporting evidence that has been authorised for distribution or use.

Release applies to an identified and immutable combination of model, profiles, documentation and release information. A released configuration should identify at least:

- Model name and version
- Topology or geometry key
- User configuration and relevant parameter values
- Material profile and version
- Process profile and version

© 2026 Magimo

- Required orientation and support strategy
- Validation and support status
- Release date and responsible creator or organisation
- Applicable documentation and handover information

Release is a configuration-control status. It does not, by itself, prove that the design is validated or suitable for every application. A configuration may be supported in principle but not yet released in a particular model revision or profile combination. Release requires final checks of the model, parameter interface, dependencies, documentation, status records and package contents. Once released, the identified combination shall be treated as immutable. Any modification requires a new release identifier, with an appropriate review of change impact and validation requirements.

6.1.5 Other status terms

- “Draft” identifies work in progress. Draft configurations may change without preserving compatibility and shall not be presented as available products or manufacturing instructions.
- “Experimental” identifies a configuration intended for controlled investigation. It may be used internally or by explicitly authorised expert testers, but it shall not be presented as a normal supported option. Experimental status should identify the purpose of the test and the known uncertainties.
- “Deprecated” identifies a configuration that remains identifiable for continuity, historic traceability or existing users but is no longer recommended for new work. Where possible, the documentation should identify the preferred replacement and explain the reason for deprecation.
- “Obsolete” identifies a configuration that is no longer maintained or authorised for new use. It may be retained as a historic record, but it shall not appear in current user-facing configuration choices.
- “Rejected” identifies a configuration that failed a requirement or is otherwise prohibited. The record should state the reason, such as a failed print trial, unacceptable dimensional behaviour, insufficient strength, incompatible material or an out-of-domain parameter combination.

These statuses should be accompanied by a reason, date and reference to the affected model, profile, configuration or evidence record.

6.1.6 Relationship between statuses

The statuses answer different questions:

- Validation: what evidence exists for the stated conditions?
- Support: is the configuration within the creator’s declared domain?
- Release: has this specific combination been authorised for distribution or use?
- Deprecation or obsolescence: is the item still recommended or maintained?

These statuses should not be treated as an automatic sequence. A configuration may be validated for a specific test condition but not supported for general use. A configuration may be supported within a declared domain while having only limited evidence. A released configuration should identify the evidence, assumptions and limitations on which the release is based.

© 2026 Magimo

6.1.7 Status and configuration selection

The user-facing configuration interface should normally expose only topology variants and options that are Supported and available in a current Released combination. Draft, Experimental, Deprecated, Obsolete and Rejected variants shall not appear as ordinary user choices.

An internal FreeCAD model may retain additional Bodies, profiles, sketches or test configurations for development and maintenance. Their presence in the model tree does not make them Supported or Released. Visibility identifies the active graphical result; it does not establish validation, support or release status. Use an explicit Configuration Key and active-result selection for those decisions. For example:

- geometryKey: TAPERED_CUBOID
- supportStatus: SUPPORTED
- releaseStatus: RELEASED

This separation prevents the model from implying that every geometry it can generate is available for normal use. See §4.6.3.

6.1.8 Status and evidence records

See Table 6–1 for definitions of the status terms.

Each status claim should be traceable to an evidence record, design decision or release decision. The record should identify the model version, parameter values or configuration key, material and process profile, orientation, test method, observed result, limitations and responsible reviewer or approver.

A status label without its associated scope and evidence is incomplete. The model, Technical File, manufacturing handover and release package should use the same terms so that the meaning remains consistent after the model leaves its creator.

6.2 Boundary testing and representative test cases

Testing should focus on the parameter boundaries and on representative combinations within the supported domain. The goal is to show that the model behaves correctly not only at the nominal point but also at the extremes and in between.

6.2.1 Test the parameter boundaries

For each key parameter (e.g. length, width, hand width, mating diameter, wall thickness):

- test the minimum supported value;
- test the maximum supported value;
- test the nominal or default value.

Where safe and practical, test just beyond a limit in a controlled development environment to confirm that the model rejects or flags the condition. Such configurations must not be treated as supported. Check that:

- the model recomputes without broken sketches or features;
- the geometry remains manufacturable under the stated process assumptions;
- critical dimensions and clearances stay within acceptable ranges; and
- any warnings or rejections behave as intended.

© 2026 Magimo

6.2.2 Test coupled extremes

Single-parameter tests are necessary but not sufficient. Parameters interact. For example:

- maximum length with minimum wall thickness may produce a flexible or unprintable part;
- maximum hand width with a small geometrical envelope may leave insufficient material;
- large hole diameter with thin surrounding walls may create a weak region; and
- certain material–orientation combinations may be marginal even if each parameter is individually acceptable.

Identify important parameter interactions and test coupled extremes, such as:

- minimum wall with maximum span;
- minimum clearance with maximum size;
- maximum load-related dimension with minimum supporting feature; and
- specific material profiles with boundary configurations.

Document which combinations have been tested and which remain outside the validated domain.

6.2.3 Test representative configurations

In addition to boundary and coupled extremes, test representative configurations that reflect realistic use cases. For example:

- a “typical” hand width with a normal fit class;
- a mid-range length with standard wall thickness;
- a common material (e.g. PLA or PA12) with the intended orientation;
- a configuration that matches a known application or customer requirement.

These tests demonstrate that the model works well not only at the limits but also in the middle of the domain where most users will operate.

6.2.4 Check more than recompute status

A successful recompute shows only that FreeCAD produced a result without reporting the relevant error. It does not establish that the geometry is printable, dimensionally suitable, structurally adequate, durable or appropriate for its intended use. Validation must therefore include checks beyond recompute status, such as:

- visual inspection of the printed part (layer quality, warpage, support marks);
- measurement of critical dimensions (holes, pins, clearances, overall sizes);
- assembly tests with mating parts;
- functional tests under load or use conditions; and
- where relevant, environmental tests (temperature, humidity, chemical exposure).

Bit holder: six test configurations

Case	rows_across × rows_out	Purpose	Expected status
Default	4 × 4	Representative	OK
Smallest	1 × 1	Lower bounds	OK

Widest	14 × 1	Bed width (203 mm)	OK
Too wide	15 x 1	Envelope check (217 mm)	Reject
Longest	4 × 10	Height and moment	Review
Largest	14 × 10	All limits at once (1.78 Nm)	Reject

For each case, check more than a clean recompute: the bit sockets number `rows_across × rows_out` plus one holder row, the holder fits the bed, every status cell shows the expected result, and the holder row stays 6 mm from the rear face.

6.3 Print trials and functional evidence

Print trials are a core part of validating a parametric DfAM product. They provide evidence that the model produces parts that are not only geometrically valid but also manufacturable and functional.

The evidence obtained from a print trial is limited to the requirements and conditions actually assessed. A successful print does not by itself establish structural suitability, dimensional suitability, durability or fitness for every intended use.

6.3.1 Record the manufacturing conditions

For each print trial, record:

- printer model and type (e.g. FDM, SLS, MJF);
- nozzle diameter (if applicable);
- layer height;
- material family and grade (including batch or supplier if relevant);
- print orientation (e.g. “Face A on build plate”);
- support strategy and infill (if relevant);
- key process settings (temperatures, speeds – at least at a summary level); and
- any deviations from the standard process profile.

This information links the test result to a specific manufacturing context. Without it, the evidence is difficult to interpret or reuse.

6.3.2 Record observations and failures

For each printed configuration, record:

- whether the part printed successfully;
- any print defects (warping, stringing, layer shifts, support marks, etc.);
- dimensional observations (e.g. holes undersized, features oversized);
- assembly behaviour (e.g. fits too tight or too loose, parts difficult to assemble);
- functional performance (e.g. stiffness, strength, movement, wear); and
- any failures or unexpected behaviour.

Even negative results are valuable. They help define the boundaries of the supported domain and inform future design changes.

© 2026 Magimo

6.3.3 Measure critical dimensions

For configurations that are intended to be functional or interface with other parts, measure critical dimensions:

- hole diameters and positions;
- pin or boss diameters;
- clearances and gaps;
- overall lengths, widths and heights;
- wall thicknesses at critical locations; and
- any other dimensions that affect fit, function or safety.

Compare the measured values with the nominal CAD dimensions and with the expected process tolerances. Document:

- the measurement method (calipers, micrometre, CMM, etc.);
- the number of samples measured;
- the observed variation; and
- whether the results are acceptable for the intended application.

6.3.4 Test function under defined conditions

Where the part has a functional role (e.g. grip, bracket, hinge, mount), test it under defined conditions that reflect the intended use:

- apply representative loads (static and, if relevant, dynamic);
- check for deformation, cracking or permanent set;
- verify that moving parts move as intended without binding or excessive play;
- assess comfort, ergonomics or usability where relevant; and
- test under environmental conditions that match the intended service conditions (temperature, humidity, chemical exposure).

Record:

- the test setup and procedure;
- the loads, displacements or other test parameters;
- the observed behaviour and any failures; and
- whether the results meet the design requirements.

6.3.5 Connect evidence to the supported domain

Evidence should be connected explicitly to the supported configuration domain. A test performed on one material, printer, orientation, size or process profile should not automatically be treated as evidence for all other configurations.

- identify which parameter configurations were tested;
- link each test to the material and process profile used;
- state whether the test result supports, limits or contradicts the claimed domain; and
- update the supported domain and documentation accordingly.

© 2026 Magimo

If a test reveals that a certain combination is marginal or unsafe, restrict the supported domain or add explicit warnings. Do not ignore evidence that conflicts with the intended claims.

> **Bit holder: two tests turn assumptions into evidence**
>
> 1. Retention and insertion: fill a holder with bits, tilt it 90°, then turn it over above a tray and shake it gently.
> Pass: no bit falls out, and every bit comes out by hand. Measure the insertion force with a spring scale; Appendix D uses the result for the misuse case.
> 2. Creep: hang the longest configuration, loaded to e.g. 2× its rated mass, on a cleat at e.g. 40 °C for 7 days. Measure the forward tilt and inspect the hook root.
> Pass: e.g. tilt change below 1 mm, no white stress marks. The result sets `allowable_moment`.
>
> Record the conditions with each result (§6.3.1): printer, material and lot, drying, orientation, perimeters, infill.

6.3.6 Tolerance-ladder coupon

A tolerance-ladder coupon is a small printed test part containing a controlled series of related dimensions, such as hole diameters, clearances, slot widths or feature gaps. It converts a clearance or fit assumption into measurable evidence for a defined material, process and orientation.

A coupon is useful when the required clearance cannot be derived reliably from first principles. Instead of selecting one assumed value, the creator prints several controlled values, measures the functional result and selects the value that gives the required behaviour.

6.3.6.1 Typical use

A tolerance ladder can be used to determine:

- clearance for a sliding fit;
- clearance for a press fit;
- hole size for a screw, pin, bearing or heat-set insert;
- pocket size for a nut or magnet;
- gap for a snap-fit or clip;
- clearance for a printed-in-place mechanism;
- dimensional compensation for a specific material and orientation.

The coupon does not validate a complete product. It provides evidence for one material-process-geometry interaction.

6.3.6.2 Basic structure

A simple clearance ladder contains several features with controlled incremental differences. For a hole-clearance ladder:

```
Clearance_n = ClearanceStart + (n - 1) * ClearanceStep
```

Table 6–2: Clearance ladder (example)

Feature	Clearance
1	0.10 mm

Feature	Clearance
2	0.15 mm
3	0.20 mm
4	0.25 mm
5	0.30 mm

Each feature should be individually identifiable. Use a number, text label or visible marker so that the measured result can be linked to the intended clearance. The coupon should also record:

Table 6–3: Coupon record fields

Field	Example
CouponID	CLR-LADDER-01
MaterialProfileKey	PETG_STD_01
ProcessProfileKey	NOZZLE_04
OrientationKey	FLAT_BASE
FeatureType	CLEARANCE_HOLE
NominalFeatureSize	5 mm
ClearanceStart	0.10 mm
ClearanceStep	0.05 mm
MeasuredResult	Insertion force, retained clearance or dimensional change
AcceptedValue	The clearance selected for release
EvidenceReference	Test record or measurement file

6.3.6.3 What to measure

The measured result must match the intended function. Do not measure only the printed dimension if the functional requirement is assembly force, retention, sliding behaviour or sealing. Examples include:

- insertion force for a press fit;
- free movement or backlash for a sliding fit;
- pull-out force for an insert or magnet;
- retention force for a snap-fit;
- leakage or sealing performance;
- measured hole, slot or pocket dimension;
- visible surface quality or edge quality.

Record both the intended value and the observed result. A feature may print dimensionally close to its target but still fail the functional requirement.

6.3.6.4 Connecting the result to the model

The accepted value becomes an input to the material, process or interface profile. It should not remain only in a test report. For example, if a 0.20 mm clearance gives the required sliding behaviour, the model may define:

```
ActiveClearance = 0.20mm [6]
```

That value should then drive the relevant geometry through the creator-control layer. The validation record should identify the coupon, material profile, process profile, orientation and test conditions that support the accepted value.

6.3.6.5 Scope of the evidence

A tolerance-ladder result applies to the combination that was tested:

- coupon geometry;
- feature type;
- material profile;
- process profile;
- print orientation;
- measurement method;
- acceptance criterion.

Do not generalise a result beyond that combination without additional evidence. A clearance validated for one material, nozzle, layer height or orientation may not be valid for another.

6.3.6.6 Practical rules

- Use a coupon when a clearance or fit value cannot be derived reliably from existing evidence.
- Include several controlled values rather than testing only one assumption.
- Make every feature identifiable.
- Record the material, process, orientation and feature type with the result.
- Measure the functional behaviour, not only the printed dimension.
- Select an accepted value based on the declared functional requirement.
- Transfer the accepted value into the controlled material, process or interface profile.
- Repeat the coupon when the material, process, orientation or feature geometry changes.

> A tolerance ladder turns an assumed clearance into measured evidence for a defined material, process, orientation and functional requirement.

---

6 Units in FreeCAD expressions:
In prose, tables and display names, this guide normally writes a value and unit with a space, for example “10 mm”. In FreeCAD Spreadsheet cells and expressions, the FreeCAD documentation recommends the compact form without a space, for example 10mm. Formulas in this guide therefore use the compact form.

Bit holder: tolerance ladder test

A strip of hex sockets from 6.40 to 6.75 mm across flats, in steps of 0.05 mm, printed in the chosen material and orientation. Test each with a real bit and pick the size that gives a normal fit: bits stay in when the holder is tilted, yet come out by hand. That size sets `socket_clearance` (See §5.3). Print one coupon per material and per socket orientation; values for vertical holes do not carry over to horizontal ones. The planned tests and their status are listed in Appendix D.5.

6.4 Error handling and controlled failure

Not every invalid input can be prevented by limits and rules alone. Users may ignore warnings, select unusual combinations or operate outside the stated conditions. The model should therefore be designed to handle errors in a controlled way.

6.4.1 Four controlled responses

A robust model can respond to invalid or marginal configurations in four main ways:

- Prevent – the model does not allow the configuration (e.g. limits, locked parameters or constraints that cannot be satisfied).
- Warn – the model generates the geometry but issues a clear warning that the configuration is outside the supported domain or may be unsafe.
- Flag – the model produces a diagnostic output (e.g. a “status” cell that reads “FAIL” or a visible marker in the geometry) that indicates a problem.
- Fail safely – the model fails to build or produces an obviously incomplete result, making it clear that something is wrong rather than silently generating nonsense.

The appropriate response depends on the severity of the issue and the consequences of misuse. Safety-critical limits should generally be prevented or cause a safe failure. Marginal configurations may be allowed with a clear warning.

6.4.2 Detect errors at the parameter layer

Where possible, detect errors at the parameter layer, before they reach the geometry. For example:

- check that wall thickness remains above the minimum for the selected nozzle and material;
- verify that clearances are positive and within acceptable ranges;
- ensure that feature counts and spacings are compatible with the available length or perimeter; and
- flag combinations that are known to be problematic (e.g. large holes with thin walls).

Parameter-level checks are easier to implement and interpret than geometry-level failures. They also allow the model to provide actionable feedback to the user.

6.4.3 Do not hide partial failure

Avoid designs that partially fail in a hidden way. For example:

- a sketch that solves but produces degenerate geometry;
- a feature that builds but omits critical details; or
- an expression that returns a default value without indicating that the input was invalid.

© 2026 Magimo

Partial failures can mislead the user into thinking that the configuration is acceptable. It is better to:

- prevent the invalid configuration where the consequence is severe;
- provide a clear warning where controlled use may be acceptable;
- flag the configuration for human review where the model cannot decide automatically; or
- stop the build or export when a valid result cannot be established.

6.4.4 Make messages actionable

Warnings and error messages should be actionable to help the user correct the problem rather than guessing what went wrong. For instance, instead of “Invalid configuration” use: “The requested wall thickness is below the supported minimum for the selected material and process profile. Increase the wall thickness or select a supported configuration.”

As a general principle, avoid any similar generic example and replace it with an actionable message that identifies:

- the affected parameter or condition;
- the reason for the warning or rejection;
- the required corrective action; and
- whether the result may be reviewed or must be rejected.

6.4.5 Controlled failure and human review

Some configurations may be too complex or unusual to handle automatically. In such cases:

- flag the configuration as requiring human review;
- provide enough information for a competent person to assess the risk; and
- document the outcome of the review.

Controlled failure combined with human review is preferable to silent corruption or uncontrolled failure.

> **Bit holder**
>
> Each response says what is wrong and what to change:
>
> - Reject: “Reject: the holder does not fit your printer – reduce the number of rows.” The Result sheet shows the width next to it and the maximum number of bits across (14 for a 220 mm bed).
> - Review: “Review: more than 6 bit rows from the wall is not yet supported.”
> - Accept: “OK: configuration supported.”
> - Reject: “Reject: the holder is too heavy for the cleat.”

6.5 Configuration management and change control

A parametric product is not a static file. It evolves over time as new materials, printers and requirements emerge. Configuration management ensures that changes are controlled, documented and compatible with earlier releases where necessary.

6.5.1 Identify the configuration elements

A released parametric product consists of multiple configuration elements:

- the FreeCAD file(s) with the model, Spreadsheets and expressions;
- material profiles and process profiles;
- validation records (test cases, measurements, functional results);
- documentation (use instructions, safety notes, manufacturing manifest);
- version identifiers for each of the above; and
- any external files or references (e.g. linked Spreadsheets, platform data).

Identify these elements and assign version identifiers to each. This makes it possible to track what was released and to reproduce a specific configuration later.

6.5.2 Version the model according to behaviour

Changes should be classified according to their effect on the model's behaviour, interface, supported domain, evidence and release status. A change that alters geometry, parameter meaning, parameter identifiers, manufacturing assumptions, validation evidence or user-visible behaviour may require a new model or guide revision.

Document what changed between versions and how it affects the supported domain and user behaviour.

6.5.3 Preserve parameter compatibility

Where possible, preserve parameter compatibility between versions:

- do not rename or remove user-facing parameters without a good reason;
- if a parameter must change, provide a mapping or migration note;
- keep the meaning of parameters consistent (e.g. `hand_width` should always mean the same thing); and
- avoid changing units or default values in a way that breaks existing configurations.

If compatibility cannot be preserved, make this explicit in the release notes and provide guidance for users upgrading from earlier versions.

6.5.4 Manage profiles separately

Material and process profiles may evolve independently of the core model. For example:

- new materials may be added;
- existing profiles may be updated with better data;
- some profiles may be deprecated as materials become unavailable or new evidence emerges.

Manage profiles as separate versioned entities:

- assign version identifiers to each profile;
- record which model versions each profile is compatible with;
- document changes to profiles and their impact on the supported domain; and
- where relevant, provide migration guidance (e.g. "Profile V2 supersedes V1; use V2 for all new configurations").

© 2026 Magimo

6.5.5 Use a compatibility matrix

For complex products with multiple model versions, material profiles and process profiles, a compatibility matrix can be useful. The matrix shows:

- which model versions are compatible with which profiles;
- which configurations have been validated for each combination; and
- any known restrictions or warnings.

This helps users and manufacturers select a valid combination and avoids unsupported pairings.

6.5.6 Perform change-impact analysis

Before releasing a changed version, perform a change-impact analysis:

- identify which parameters, expressions or geometry features have changed;
- assess how these changes affect the supported domain;
- determine whether existing configurations remain valid;
- identify any new risks or failure modes; and
- decide whether additional testing or validation is required.

Document the results of the analysis and any actions taken (e.g. additional tests, updated documentation, restricted domain).

6.5.7 Release immutable combinations

A released combination should identify the exact model version, parameter schema, manufacturing profile, documentation and supporting evidence. Once released, that combination should not be changed silently. Changes should produce a new identifiable release or revision. This preserves traceability and makes it possible to reproduce a specific configuration later, for example in case of a safety investigation or customer query.

6.6 In-model documentation

6.6.1 Document during design

Documentation should be created and maintained during the design process rather than reconstructed after the model is complete. Decisions, assumptions, parameter meanings, limitations and evidence are easier to record accurately when they are still available to the creator.

Documentation written afterwards often becomes a retrospective chore. It is more likely to be incomplete, generic or disconnected from the decisions that actually shaped the model. Documentation written during design is more useful to the creator as well as to future users: it forces important questions to be answered while they can still influence the design.

A practical rule is to add the description, unit, permitted range, source or design rationale at the same time as the parameter, expression, profile or validation record. If a decision is too uncertain to document clearly, that is often a useful signal that the design decision itself needs more thought.

6.6.2 Document parameters where they live

Each parameter in the Spreadsheet or parameter layer should have:

© 2026 Magimo

- a meaningful name or alias;
- a short description of what it represents;
- the expected unit;
- the supported range or options;
- the default value; and
- any relevant notes (e.g. “Do not change manually”, “Derived from `fit_class` and `nozzle_diameter`”).

This makes the parameter interface self-documenting and easier to audit.

6.6.3 Document protected logic

Protected engineering values and rules should also be documented, even if they are not user-editable. For example:

- minimum wall rules and their basis (nozzle, material, process);
- clearance formulas and the assumptions behind them;
- orientation requirements and the reasoning (e.g. load path, surface quality); and
- any hard rules that cause configuration rejection.

This documentation helps future maintainers understand why the model behaves as it does and prevents well-meaning but harmful changes.

6.6.4 Document the model itself

At a higher level, document the model as a whole:

- the intended function and use cases;
- the supported configuration domain;
- the parameter architecture (context, choice, engineering, derived);
- the reference strategy (datums, master sketches, key constraints);
- the expression structure and key relationships; and
- any known limitations or open issues.

This can be stored in:

- a dedicated “Docs” or “Info” sheet in the FreeCAD file;
- an external README or technical note linked from the model; or
- a platform-specific description field.

6.6.5 Document status and limitations

Be explicit about status and limitations:

- which configurations have been validated and how;
- which configurations are supported but not fully tested;
- which combinations are known to be marginal or unsupported;
- any safety-related restrictions or warnings; and

© 2026 Magimo

- the intended service conditions of use (loads, temperatures, environment).

This documentation protects both the creator and the user by making the boundaries of responsibility clear.

6.6.6 Keep documentation maintainable

Documentation should be easy to update:

- avoid duplicating information in multiple places;
- where possible, generate documentation from the model (e.g. parameter tables, configuration summaries);
- use clear, concise language;
- structure the documentation so that changes can be made without rewriting everything; and
- version the documentation along with the model and profiles.

Maintainable documentation is more likely to stay up to date and therefore more likely to be useful.

6.7 Packaging the released file

The released file is the tangible product that users and manufacturers receive. It should be clean, identifiable and complete.

6.7.1 Start from a release copy

Do not release the working development file. Instead:

- create a copy of the model specifically for release;
- remove or isolate debugging geometry, test sketches and experimental features;
- ensure that only supported parameters and profiles are exposed; and
- verify that the file recomputes cleanly at the default configuration.

This reduces the risk of accidentally exposing unfinished or unsupported functionality.

6.7.2 Clean the model tree

Review the feature tree and remove or hide:

- unused sketches, datums or Bodies;
- intermediate construction geometry that is not needed for understanding;
- duplicate or superseded features; and
- any features that exist only for testing or debugging.

A clean tree makes the model easier to understand and reduces the risk of confusion.

6.7.3 Remove or isolate debugging material

Debugging material (e.g. extra Spreadsheets, diagnostic expressions, test configurations) should be:

- removed from the released file; or

- moved to a separate “Debug” or “Internal” section that is clearly marked as not part of the released product.

If debugging material is retained for future maintenance, document its purpose and ensure that it cannot be mistaken for part of the user interface.

6.7.4 Control dependencies

Every external dependency required to open, configure, generate, validate or manufacture the released result should be identified and controlled. This includes linked files, external profiles, scripts, workbenches, macros, libraries and version-specific resources.

Each dependency should be embedded, included in the release package or referenced through a stable versioned location. A release should not depend silently on an unversioned or unavailable external file.

If external dependencies are necessary (e.g. a material database), provide them as part of the release package and document how they are used.

6.7.5 Include supporting files

A complete release package may include:

- the FreeCAD file(s);
- material and process profile definitions (if external);
- a manufacturing manifest or handover file;
- use instructions and safety notes;
- a summary of validated configurations; and
- version and compatibility information.

Package these files together in a clear structure (e.g. a zip archive or platform bundle) so that users and manufacturers receive everything they need.

6.7.6 Identify versions and status

The released file and package should clearly identify:

- the model version (e.g. “Grip_v2.1”);
- the profile versions (e.g. “MaterialProfile_PLA_v1.3”, “ProcessProfile_FDM_0.4_v2.0”);
- the documentation version;
- the release date;
- the creator or responsible entity; and
- the status (e.g. Released, Supported, Experimental).

This information can be included in:

- the file name;
- a title block or info sheet inside the FreeCAD file;
- a README or metadata file; and
- platform-specific fields if distributed through a service.

© 2026 Magimo

6.7.7 Perform the release check

Before releasing, perform a final check:

- recompute the model at the default configuration and verify that it builds cleanly;
- test at least one boundary configuration to confirm that limits and warnings work;
- review the parameter interface to ensure that only supported inputs are exposed;
- check that all documentation is present, consistent and up to date;
- verify that version identifiers and status are correct; and
- confirm that the release package contains all necessary files.

This release check is the last line of defence against errors and omissions. It should be performed systematically and documented as part of the release record.

© 2026 Magimo

7 Handover, Platforms and Design-System Thinking

This chapter covers what leaves with the model: the Technical File, the manufacturing handover and the use and safety information. It then shows how the same principles carry over to scripts, external interfaces and distribution platforms, and how a single model grows into a controlled design system.

7.1 The Technical File

The Technical File is the controlled record of the design basis, supported configuration domain, manufacturing assumptions, validation evidence, release status and known limitations. It preserves information that may not be visible from the FreeCAD model itself.

The Technical File may be maintained internally during development and reduced to released documentation for users, manufacturers or other recipients. These records should remain traceable to the applicable model, profile, configuration and release.

7.1.1 What the Technical File records

The Technical File captures the reasoning, assumptions, evidence and limitations behind the parametric product. It is not necessarily a single document; it may be a controlled set of linked files, sheets and records. Its content should remain traceable to the applicable model, profile, configuration and release.

For any released configuration, the Technical File should make it possible to identify the applicable configuration key, selected parameter values, model version, material profile, process profile, required orientation, status and evidence record.

7.1.1.1 Design intent and requirements

- intended function and use;
- intended users and use cases;
- design requirements and interfaces;
- principal loads and load directions;
- environmental and service conditions; and
- applicable regulatory, safety or customer requirements.

This section answers the question: what is this part supposed to do and under which conditions?

7.1.1.2 Assumptions and design boundaries

- assumed manufacturing processes and profiles, for example FDM with a 0.4 mm nozzle and PETG;
- materials considered, supported, restricted or excluded;
- orientation requirements and their rationale;
- the supported configuration domain, including parameter ranges and permitted combinations;
- assumptions about assembly, post-processing, inspection and maintenance; and

© 2026 Magimo

- explicit boundaries beyond which the design is not intended to be used.

This section makes the creator's assumptions visible so that they can be reviewed, challenged and updated.

7.1.1.3 Design rationale

- why key rules or parameters were chosen, for example wall thickness, clearance rules or rib spacing;
- why certain options were allowed or prohibited;
- why particular materials, orientations or process profiles are required or recommended;
- trade-offs considered, for example strength versus weight or print time versus surface quality; and
- alternative concepts explored and the reasons they were rejected.

Design rationale is often the first information lost when a model is handed over. Recording it explicitly makes later changes safer and more informed.

7.1.1.4 Parametric architecture

- parameter classification: context, choice, engineering and derived;
- the structure of the Spreadsheet or other parameter layer;
- the reference strategy, including datums, master sketches and key constraints;
- the expression structure and important relationships;
- feature-tree organisation and Body strategy; and
- manufacturing-aware rules encoded in the model.

This section helps a new engineer understand how the model works without having to reverse-engineer it from the geometry alone.

7.1.1.5 Validation evidence

- test configurations examined, including parameter values and material or process-profile combinations;
- manufacturing conditions for each test, including printer, nozzle, layer height, orientation and relevant post-processing;
- observations, measurements and functional results;
- failures or marginal behaviour and how they were addressed;
- configurations considered Validated, Supported, Experimental or otherwise outside the supported domain; and
- references to test reports, measurement logs, photographs, calculations or other evidence.

Validation evidence links the claims about the supported domain to actual data.

7.1.1.6 Limitations and open issues

- known design limitations, such as unsupported configurations or marginal materials;

© 2026 Magimo

- open questions or assumptions not yet fully validated;
- known risks and their mitigation, for example “Do not use in high-temperature environments”;
- safety-related restrictions or warnings; and
- planned improvements or future work.

Explicit limitations help users and maintainers understand where the design boundary lies and where further work is needed.

7.1.1.7 Revision history

- version identifiers for the model, profiles, documentation and associated evidence;
- a summary of changes in each version;
- the date and responsible author, reviewer or approver for each change;
- the impact of each change on the supported domain, compatibility, evidence and release status; and
- references to related change records, test reports, issue trackers or release records.

A clear revision history makes it possible to trace how the product evolved and to understand why a particular version behaves as it does.

7.1.2 Internal record versus released documentation

The internal Technical File may contain development notes, failed tests, experimental configurations, alternative designs and detailed engineering calculations. Released documentation should contain the information needed by its intended recipient to configure, manufacture, inspect and use the result correctly. It need not expose every internal development record, but it must not omit limitations that affect safe or correct use.

© 2026 Magimo

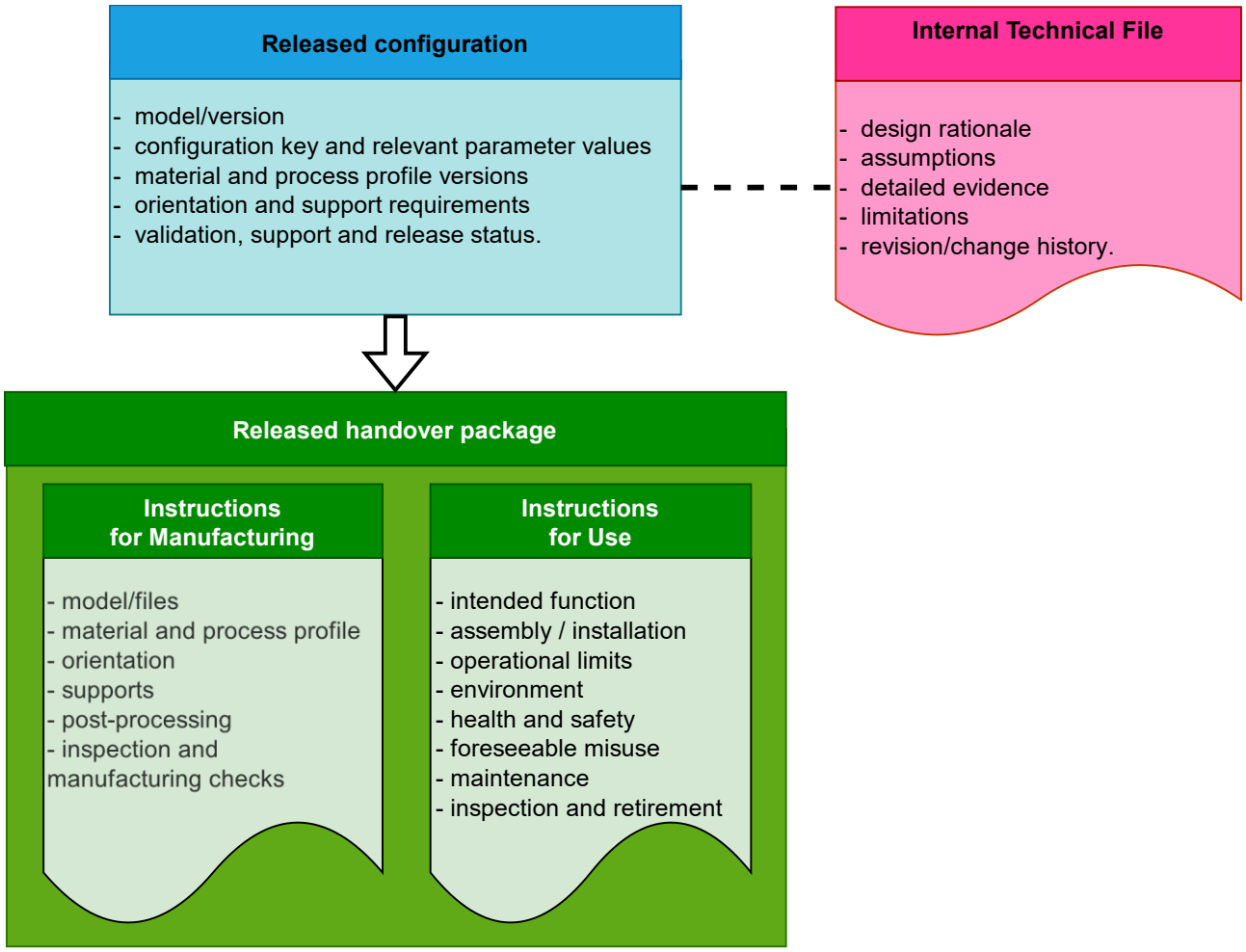


Figure 7–1: Relationship between the internal Technical File and the released handover package

The released package is a controlled, use-specific subset of the engineering record, tied to an identified model, configuration and material/process profile combination. It must remain consistent with the internal evidence and limitations.

> **Bit holder**
>
> The design basis, calculations, assumptions and open issues of the bit holder are collected as a Technical File extract in Appendix D. It shows what §7.1.1 asks for on one real part, including the issues that are not yet resolved.

7.2 The manufacturing handover

The manufacturing handover transfers the information needed to produce the configured part under the conditions on which the design is based. For a 3D-printed part, a generated geometry file alone is not enough: orientation, material, process settings, support strategy, post-processing and inspection all affect the result.

This section covers the handover for parametric 3D-printed parts. Machined and sheet-metal parts are usually handed over as toleranced drawings and are not discussed separately; the same principle applies, but the drawing carries the information.

© 2026 Magimo

7.2.1 Material and printer

Identify the required or permitted material and the manufacturing equipment or process profile that form the design basis. Where alternatives are supported, identify them explicitly rather than implying that all materials or printers are interchangeable. Specify:

- the required or permitted material family and grade (e.g. “PETG, FDM” or “ASA, FDM”);
- any prohibited materials (e.g. “Do not use PETG for this application”);
- the printer type and, where relevant, the specific printer model or class;
- nozzle diameter and type (for extrusion-based processes);
- permitted or required layer-height range;
- any other process-specific requirements (e.g. enclosure, heated bed).

If material or process profiles are used, reference them explicitly, for example for a laser-cut part: “Use MaterialProfile_Plywood_3mm_v1.2 and ProcessProfile_LaserCut_CO2_v2”.

7.2.2 Orientation and load direction

State the required or permitted build orientation:

- which face should be on the build plate (e.g. “Face A on build plate”);
- the principal load direction and its relationship to the build orientation;
- any orientation constraints for critical features (e.g. holes, bosses, mating surfaces); and
- the consequences of changing the orientation (e.g. reduced strength in Z, different support requirements).

For load-critical parts, orientation may be a hard requirement rather than a suggestion. If so, state it as a release condition.

7.2.3 Walls, shells and infill

Specify:

- the required or assumed wall thickness and number of perimeters, where relevant;
- the required or assumed top- and bottom-shell thickness;
- the infill type and density, where they affect stiffness, weight, function or validation assumptions;
- any local reinforcements (e.g. ribs, gussets) and their purpose; and
- any restrictions on modifying these settings, for example: “Do not reduce infill below 40% for load-bearing configurations.”

If the model already encodes these values in the geometry, the handover should still state the relevant assumptions so that the manufacturer understands the design basis.

7.2.4 Supports, cavities and seams

State the requirements or guidance for:

- where supports are expected or permitted;
- where supports must not be placed (e.g. on mating or bearing surfaces);

© 2026 Magimo

- how to handle internal cavities, for example avoiding trapped support material and inaccessible volumes;
- seam-placement preferences, where relevant to appearance, strength, sealing or function;
- any post-processing related to supports (e.g. careful removal from marked surfaces, sanding instructions).

Clear support instructions reduce the risk of damage during support removal and help achieve the required surface quality on critical faces.

7.2.5 Fits, tolerances and scaling

State:

- the intended fit classes for key interfaces (e.g. “Hole-pin fit: normal”);
- the expected dimensional tolerances for the stated process, material and relevant feature directions;
- any directional compensation or other process compensation applied in the model;
- whether additional post-processing is required or permitted, for example reaming holes or sanding surfaces; and
- any assembly instructions related to fits (e.g. “Assemble by hand; do not force”).

If the model includes multiple fit options, identify the option that applies to the released configuration.

7.2.6 Compact manufacturing manifest

A compact manufacturing manifest summarises the key manufacturing information in a structured, machine-readable or easily parsable form. It may include:

- model identifier and version;
- configuration key and selected parameter values or the name and version of the applicable preset;
- material profile identifier and version;
- process profile identifier and version;
- orientation requirement;
- support strategy;
- infill and shell settings (if relevant);
- fit and tolerance assumptions;
- post-processing requirements; and
- any special notes or warnings.

The manifest may be provided as:

- a dedicated sheet in the FreeCAD file;
- a separate text, JSON or CSV file;
- a platform-specific data structure; or
- part of a larger production order or job file.

Its purpose is to make the manufacturing intent explicit and portable.

© 2026 Magimo

Bit holder – manufacturing manifest (example)

- Model: bit_holder v1.0
- Configuration: 4 × 4 with the holder row, 25 mm bits, normal fit
- Material profile: PETG v1.0 (dry before printing)
- Process profile: 0.4 mm nozzle, 0.2 mm layers, v1.0
- Orientation: top face on the bed
- Supports: none
- Shells and infill: 4 perimeters, 20 % infill
- Fit assumptions: socket clearances from the PETG coupon, vertical sockets
- Post-processing: none; do not scale
- Notes: first-part check: insert three bits (they must hold and come out by hand); hang the holder on a cleat offcut (it must sit flat against the wall)

7.3 The use and safety handover

The use and safety handover informs the end user, installer or operator how to install, use, inspect and retire the part safely and effectively.

7.3.1 Structural limits

State:

- the maximum stated static load, force, mass or pressure under the specified support, orientation, temperature, duration and application conditions;
- the maximum stated dynamic or cyclic load and applicable cycle or duration limits, if relevant;
- the load direction or directions and application points for which the part is validated;
- any restrictions on load application points or lever arms;
- the safety factors or design margins applied, where they are known and appropriate to disclose; and
- the consequences of exceeding the stated limits (e.g. deformation, fracture, loss of function).

For safety-critical parts, these limits must be clear, conservative, prominently communicated and supported by application-specific engineering analysis and validation.

7.3.2 Environmental and chemical limits

Specify:

- the allowable temperature range, including minimum and maximum operating temperatures and any duration limits;
- any restrictions on exposure to UV, moisture, chemicals or solvents;
- the expected service life or service-life assumptions under the stated conditions;
- any ageing or degradation mechanisms that are relevant (e.g. creep, UV embrittlement); and
- storage and handling recommendations (e.g. keep dry, avoid direct sunlight).

© 2026 Magimo

If the part is intended for a special environment, such as food contact, medical use or outdoor service, state the applicable certifications, restrictions, qualifications or additional requirements. Do not imply compliance merely because the material or process is commonly used in that environment.

7.3.3 Assembly and installation

Provide instructions on:

- how the part is to be assembled (e.g. order of operations, torque values for fasteners, adhesive curing conditions);
- any tools or fixtures required;
- alignment or orientation requirements during installation;
- any adjustments, calibrations or checks required before use; and
- common mistakes to avoid (e.g. over-tightening, incorrect orientation, forcing fits).

Clear assembly instructions reduce the risk of installation errors that could compromise function, durability or safety.

7.3.4 Foreseeable misuse

Consider reasonably foreseeable misuse and address scenarios explicitly where they could create a significant risk of failure, damage or injury:

- using the part outside the stated load or temperature limits;
- using it with incompatible materials or in unsupported orientations;
- modifying the part (e.g. drilling additional holes, cutting, sanding critical surfaces);
- using it for applications other than those intended (e.g. using a desk accessory as a structural bracket); and
- ignoring warnings or instructions.

For each significant foreseeable misuse, consider whether to:

- design the part or interface to make the misuse difficult or impossible; or
- state clearly in the documentation that the misuse is prohibited and explain the relevant risks.

7.3.5 Inspection and retirement

Provide guidance on:

- how often the part should be inspected, for example before each use, monthly or annually, according to the intended service conditions;
- what to look for during inspection (e.g. cracks, deformation, wear, discoloration);
- criteria for retirement or replacement, for example visible cracks, permanent deformation, excessive wear or loss of fit;
- any maintenance actions that are required or recommended (e.g. cleaning, lubrication); and
- how to dispose of the part at end of life (e.g. recycling, special disposal for certain materials).

For safety-critical or heavily loaded parts, inspection and retirement criteria are essential and should be based on the applicable risk and service conditions.

© 2026 Magimo

Bit holder – use instructions

- Structural limits: screw bits only, up to the configured number.
- Environment: indoors, below e.g. 40 °C, out of direct sun.
- Installation: hang on a 45° cleat of 18 mm plywood; leave e.g. 30 mm free above the holder so it can be lifted off.
- Foreseeable misuse (§7.3.4): pushing bits in while the holder hangs on the wall; hanging other tools from it; pulling it off the wall by the outer row. The instructions name each one.
- Inspection and retirement (§7.3.5): check the hook root for cracks or white stress marks; replace the holder when it tilts forwards on the cleat or bits no longer stay in.

7.4 External parameter interfaces and automation

Parametric models are increasingly used in automated workflows: web configurators, scripts, platform APIs and generative systems. External interfaces must use the parameters defined in the model and comply with its validation rules.

7.4.1 Preserve the parameter roles

External interfaces should preserve the roles of parameters:

- context parameters, such as printer, material and orientation, should remain contextual inputs and should not be turned into arbitrary geometry drivers;
- choice parameters (e.g. fit class, pattern type) should remain discrete, validated options;
- engineering parameters, such as minimum-wall rules and clearance rules, should remain under creator control; and
- derived parameters (e.g. wall thickness, inner diameter) should remain outputs, not independent inputs.

Do not expose every Spreadsheet alias as a public field. The external interface should reflect the same responsibility boundary as the internal model.

7.4.2 Define an interface contract

Treat the external parameter interface as a contract:

- define each input, including its stable name, type, unit, range and permitted options;
- define each output, including its meaning, unit and dependency;
- specify the expected behaviour for valid, warned, rejected and failed inputs;
- document how errors and warnings are communicated;
- state which configurations are Supported and which are rejected; and
- version the interface so that changes can be tracked.

This contract should align with the internal parameter contract described in earlier chapters.

7.4.3 Validate before generation

Validate inputs and known parameter combinations before geometry generation where practical. Perform geometry-dependent checks after generation but before the result is released or exported.

- check that all inputs are within their defined ranges;

- verify that parameter combinations are allowed;
- apply the same hard rules and soft guidance as the internal model;
- reject or flag known-invalid configurations before calling the CAD engine where practical;
- perform geometry-dependent checks after generation; and
- provide clear, actionable feedback to the user.

This prevents the CAD system from being treated as a black box and ensures that generated geometry is assessed against the design rules before release.

7.4.4 Handle formats carefully

When exchanging parameter data between systems (for example: web form → script → FreeCAD):

- use consistent units (e.g. always mm, not mixed mm/inch);
- avoid locale-dependent number formats (e.g. decimal comma vs point);
- use stable identifiers for strings such as material profiles and fit classes, rather than relying only on display names;
- handle missing or out-of-range values gracefully; and
- log input data and generated configurations for traceability.

Small format mismatches can cause large errors if they are not detected and handled carefully.

7.4.5 Keep generation deterministic

Geometry generation should be deterministic:

- the same model version, profile versions, inputs and generation environment should produce the same outputs within the defined numerical tolerance;
- avoid random or time-dependent elements unless explicitly intended and documented;
- ensure that the CAD engine is called with consistent settings; and
- version the model and profiles so that the same interface call can be reproduced later.

Deterministic generation supports quality control, debugging, comparison and traceability.

7.4.6 Separate inputs from outputs

The external interface should clearly separate:

- inputs – values provided by the user or system, such as hand width, fit class or material profile;
- outputs – values calculated by the model, such as wall thickness, clearance or weight; and
- status – information about whether the configuration is accepted, warned, rejected, Supported, Validated or Released, as applicable.

Do not allow users to overwrite outputs or status fields. They are results, not inputs.

7.4.7 Preserve the design intent during automation

Automation should not bypass the parameter contract. For example:

- do not generate configurations that the model explicitly rejects;
- do not hide warnings or errors to “make the system look better”;

© 2026 Magimo

- do not change the meaning of parameters or profiles without updating the interface contract; and
- do not claim validation or support beyond the configurations and conditions covered by the available evidence.

The automation layer is an extension of the design system, not a shortcut around it.

7.4.8 External parameter libraries

An external parameter library is a controlled Spreadsheet or document that supplies shared values to more than one model. It can hold material data, process profiles, hardware dimensions, standard sizes, clearance rules or common engineering constants.

A shared library reduces duplication, but it also introduces a dependency on another file. The library must therefore be treated as a controlled part of the parameter contract, not as a convenient external cell reference.

7.4.8.1 When to use one

An external library is useful when several models must use the same controlled values, for example:

- material properties and compensation rules;
- nozzle, layer-height and clearance profiles;
- standard hardware dimensions;
- approved fastener, insert, bearing or magnet data;
- common interface dimensions;
- company-wide engineering constants.

Do not use an external library merely to move values out of the current model. If only one model uses the values, a local, clearly structured parameter sheet is usually simpler and more robust.

7.4.8.2 Reference format

FreeCAD refers to a value in another document using the master document's name, followed by the object and property:

```
master#modelConstants.MinimumWallThickness
```

The exact reference depends on the document name, object name and property or alias being used. Before creating dependent documents:

1. Create the master document.
2. Add its Spreadsheet or parameter object.
3. Save the master document.
4. Create and save the dependent documents.
5. Verify that the external references resolve correctly.

Do not rename the master document after dependent documents have been created. Renaming can break external links. Keep the master document available and loaded when its values are required by dependent documents.

7.4.8.3 Control the library

An external library should have the same discipline as a released model:

Table 7–1: External library discipline

Item	Requirement
Identity	Use a stable library identifier and revision
Ownership	Define who may change the library
Values	Give every shared value a meaningful alias
Units	State the unit for every physical value
Status	Record whether each value or profile is supported or validated
Compatibility	Record which model versions may use the library revision
Change control	Version the library when values, rules or supported combinations change
Availability	Ensure the library is available wherever dependent models are opened or generated

A shared value should have one authoritative definition. If the same value is also stored locally in every dependent model, the copies will eventually diverge.

7.4.8.4 Local result values

A dependent model may need a local calculated result based on a library value. For example:

```
LocalMinimumWall = master#materialProfiles.MinimumWallThickness
```

This is acceptable when the local value is clearly identified as a derived result. It should not become an independent second definition of the library value. If a model must continue to work when the library is unavailable, use an explicit local snapshot with its own provenance:

Table 7–2: Local snapshot of library values

Alias	Meaning
LibraryMinimumWall	Live value from the external library
SnapshotMinimumWall	Recorded value captured from an approved library revision
SnapshotSource	Library identifier and revision
SnapshotDate	Date the value was captured

A snapshot is not a live link. It must be clearly labelled, versioned and updated deliberately.

7.4.8.5 Practical rules

- Use an external library only when several models genuinely share controlled values.
- Save the master document before creating dependent documents.
- Do not rename a master document after dependent models reference it.
- Give every shared value a meaningful alias and explicit unit.

© 2026 Magimo

- Treat the library as a versioned configuration item.
- Keep one authoritative definition for each shared value.
- Record the library identifier and revision with each dependent model.
- Do not allow an unavailable or silently changed library to alter a released model without detection, review and an explicit compatibility decision.
- Use a clearly labelled local snapshot only when offline operation or controlled release requires it.

Rule:

> An external parameter library is a controlled interface between models, not a hidden source of values.

7.5 Platform-neutral distribution requirements

Whether a parametric product is distributed as a FreeCAD file, through a print service or via a platform, certain requirements remain platform-neutral.

7.5.1 Minimum release package

At minimum, a released parametric product should include:

- the parametric model (e.g. FreeCAD file) in a clean, released state;
- a parameter schema or interface definition (names, types, units, ranges, options);
- manufacturing information (material and process assumptions, orientation, supports, fits);
- use and safety information (loads, limits, environmental conditions of use, assembly instructions);
- version identifiers for the model, profiles and documentation;
- the configuration key, selected parameter values and required dependencies; and
- a statement of the supported configuration domain and validation status.

Additional elements, such as the Technical File and detailed test reports, may be retained internally or provided on request, but the minimum package must be sufficient for a competent user or manufacturer to configure, produce, inspect and use the part within the stated limits.

7.5.2 The parameter schema is part of the product

The parameter schema – the structured definition of inputs, outputs and rules – is part of the product, not an implementation detail. It defines:

- what the user can configure;
- what the system controls;
- what combinations are allowed; and
- how the model behaves.

When distributing a parametric product, the schema should be:

- documented in a stable, versioned form;
- consistent with the internal model; and

- portable across platforms, for example expressible in JSON, CSV or another suitable neutral format where needed.

7.5.3 Minimum operational behaviour

A released parametric product should exhibit minimum operational behaviour:

- it recomputes without relevant errors at the default configuration;
- it enforces hard rules (e.g. rejects or flags configurations that violate minimum wall, maximum overhang or other non-negotiable limits);
- it provides meaningful warnings for marginal or unsupported configurations;
- it produces geometry that is checked against the stated manufacturing assumptions; and
- it fails in a controlled way when inputs are invalid or inconsistent.

If a model does not meet these criteria, it is not yet a released product, regardless of how clever its geometry may be.

7.5.4 Separate creator and user packages

Where practical, separate:

- the creator package – the full development environment, including debugging tools, internal notes and experimental features; and
- the user package – the clean, released product with only supported parameters, profiles and documentation.

This separation keeps unfinished or unsupported functionality out of the normal user workflow while allowing creators to continue developing and testing new ideas. See also §3.1.4.

7.6 A distribution platform implementation

This section is a brief note on how the principles in this guide map to a distribution platform. It is not a complete platform specification; it outlines the key alignment points.

7.6.1 Generic concept to platform mapping

The generic concepts in this guide map to platform elements as follows:

- User configuration layer → platform configuration form (input fields, selectors, presets);
- Creator control layer → protected model-generation logic (FreeCAD engine, scripts, expressions);
- Material and process profiles → platform material and process definitions;
- Validation and support records → platform test, evidence and status metadata;
- Manufacturing manifest → platform job, order or production data; and
- Use and safety documentation → platform product description and safety notes.

The mapping should preserve the parameter roles, responsibility boundaries, configuration key, validation evidence, support status and release status.

© 2026 Magimo

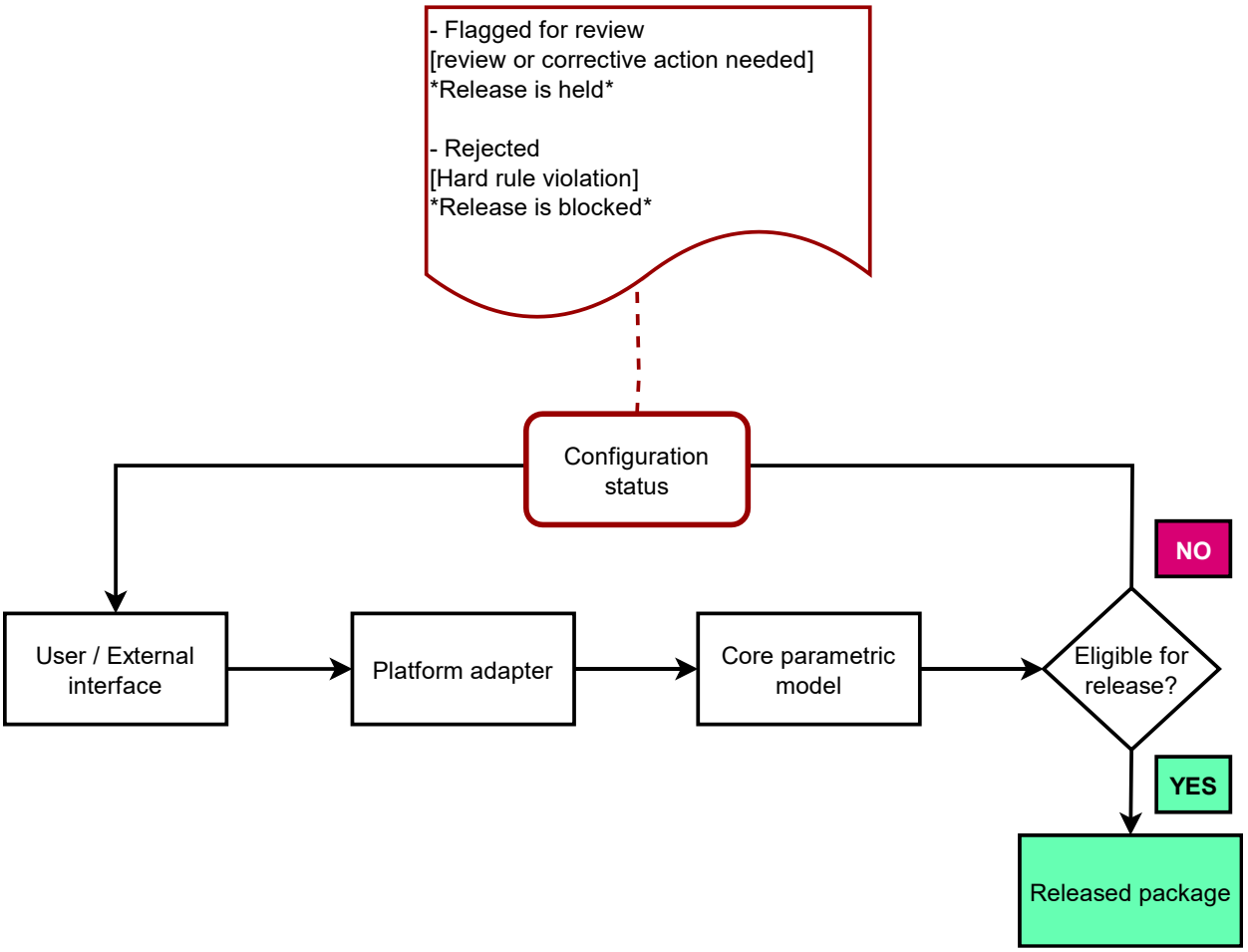


Figure 7–2: Platform adapter around a platform-neutral parametric engineering model

The adapter maps the external interface to the model's stable parameter contract; the core model retains geometry-generation and validation authority and returns both the result and its status. The platform presents those outputs and records the released configuration.

7.6.2 Use stable identifiers

On a platform, use stable identifiers for:

- model versions;
- material and process profiles;
- configuration presets;
- configuration keys; and
- evidence or release records.

Stable identifiers help to:

- reproduce a specific configuration later;
- trace issues back to a specific version; and
- manage compatibility across updates.

Avoid relying on display names alone, because they may change without changing the underlying identifier.

© 2026 Magimo

7.6.3 Keep validation in the model boundary

Validation authority should remain in the model boundary rather than in the platform UI alone. The platform may perform preliminary input checks, but the model boundary or a demonstrably equivalent controlled service must determine whether a configuration is accepted, warned, rejected, Supported, Validated or Released. Hence:

- present the validation and support results to the user, for example accepted, warned or rejected;
- prevent generation of rejected configurations;
- display warnings for marginal or unsupported configurations;
- record the model, profile, configuration and status with each generated job; and
- preserve the relevant evidence or evidence reference.

The platform should refrain from bypassing or altering the model's validation rules unless the design intent is intentionally modified.

7.6.4 Preserve status and evidence

The platform should preserve:

- the applicable validation, support and release status of each configuration;
- the material and profile versions used;
- the manufacturing parameters, including orientation, supports, material and process profile; and
- any user-visible warnings, limitations or notes.

This information may be needed for:

- customer support;
- quality investigations;
- regulatory compliance; and
- future design improvements.

7.6.5 Platform-specific material belongs in an adapter layer

Platform-specific logic, such as UI conventions, API details, pricing, marketplace metadata and workflow state, should live in an adapter layer rather than in the core engineering model. The core model should:

- remain platform-neutral in its engineering logic;
- expose a clear parameter interface; and
- be callable by different platforms or scripts without requiring changes to its engineering meaning.

This keeps the engineering model maintainable and portable while allowing each platform to handle its own operational requirements.

© 2026 Magimo

7.7 From shape to controlled design system

A parametric DfAM product is not merely a shape that can be scaled or adjusted. It is a controlled design system that encodes:

- design intent and requirements;
- parameter architecture and responsibility boundaries;
- manufacturing knowledge and constraints;
- validation evidence and limitations; and
- handover information for manufacturers and users.

The shift from “shape” to “controlled design system” is a central message of this guide. It requires:

- disciplined parameter design;
- robust modelling practices;
- explicit manufacturing assumptions;
- systematic validation; and
- clear documentation and handover.

When these elements are in place, parametric design becomes a more reliable engineering tool rather than a source of hidden risks and unsupported configurations.

© 2026 Magimo

Appendix A Minimum checklist for simple shared parts

A.1 Purpose

Not every model needs the full process in this guide. A simple part shared with others, such as a bracket, a clip or a holder, can be released responsibly with a short record. This appendix gives the minimum: what you must decide, record and check before someone else configures or prints your model. Use this checklist when all of the following are true:

- the part has one main function and few parameters (about five or fewer exposed)
- failure would cause inconvenience, not damage or injury
- you are not making claims about strength, food contact, heat or chemical resistance

If any of these is not true, use the full process (Chapters 2–7).

A.2 Checklist

Table A–1: Minimum checklist

Item	What to record	Example (cable clip)	Done	N/A
Function	What the part does, in one sentence	Holds one cable against a desk edge	☐	☐
Not for	Uses you do not support	Not for load-bearing or outdoor use	☐	☐
Fixed dimensions	Dimensions the user must not change, and why	Clip opening angle: sets the snap force	☐	☐
Variable dimensions	Each exposed parameter: name, unit, meaning	cable_diameter (mm): measured outside diameter	☐	☐
Supported range	Minimum and maximum for each exposed parameter, and any combination that does not work	cable_diameter 3–10 mm; above 8 mm use the wide variant	☐	☐
Material	Material(s) the design was made for; materials to avoid	PETG; not PLA (creeps under constant clamping force)	☐	☐
Orientation	Required print orientation and the reason	Flat side on the build plate: the clip flexes across the layers	☐	☐
Process assumptions	Nozzle, layer height, walls; the settings the design relies on	0.4 mm nozzle, 0.2 mm layers, at least 3 walls	☐	☐
Default printed once	The default configuration has been printed and checked for fit and function	Printed in PETG, fitted on a 6 mm cable	☐	☐
Known limits	What you know does not work, or have not tested	Ranges below 4 mm not printed; snap may loosen after repeated use	☐	☐
Failure behaviour	What the user sees outside the supported range	Status cell shows "Outside supported range"	☐	☐
Version	Model version and FreeCAD version used	cable_clip v1.0, FreeCAD 1.1.2	☐	☐

© 2026 Magimo

A.3 Where to put it

Put the record where the user will see it before printing: the first rows of the user Spreadsheet, a README file shipped with the model or the description on the download page. One short record that travels with the file is better than a complete one that stays with you.

A.4 When to move to the full process

Move to the full process when the part starts to grow: more parameters, more materials, other people printing it for third parties, or a use where failure matters. Start with Chapter 2. The record you made here becomes the first draft of your design basis (§2.1).

© 2026 Magimo

Appendix B Practical FreeCAD implementation of the Spreadsheet control layer

This appendix provides a practical implementation reference for the Spreadsheet control layer introduced in Chapter 3. It supports FreeCAD models in which user configuration, creator-controlled engineering logic, manufacturing assumptions, validation evidence and release information remain distinguishable and traceable.

It does not repeat the principles of parameter roles, controlled freedom, supported configuration domains or the creator–user responsibility boundary. Refer to Chapters 2 and 3 for those principles. This appendix concentrates on practical file structure and working conventions.

B.1 Purpose and scope

Use the Spreadsheet workbench as a structured control layer between configuration inputs and geometry. The aim is not to place every value in a Spreadsheet. The aim is to prevent values with different responsibilities, sources and validation status from becoming mixed together. A suitable Spreadsheet architecture should enable a competent maintainer to identify:

- Which values are entered or selected by the user.
- Which values are controlled by the creator.
- Which values are calculated by the model.
- Which material and process assumptions apply.
- Which configuration has been tested and released.
- Which documentation and evidence belong to the model version.

The geometry should normally consume creator-controlled or derived values rather than raw user inputs. This keeps the parameter interface separate from the engineering logic that interprets it.

B.2 Minimum sheet structure

For a distribution-ready parametric model, the following sheets provide a practical starting structure. Apply the structure proportionately: a simple internal model may combine some functions or omit sheets that serve no real purpose.

Table B–1: Recommended sheet structure

Sheet	Primary purpose	Typical editor
“UIParams”	User-facing configuration values and deliberate choices	User or configurator
“CMParams”	Creator-controlled engineering values, calculations and diagnostics	Creator or maintainer
“MaterialProfile”	Material-property data, compensation factors and qualification status	Creator or materials owner
“ProcessAspects”	Process assumptions, manufacturing obligations and process limits	Creator or manufacturing owner
“UseConstraints”	Intended use, service conditions, limitations and foreseeable misuse	Creator or product owner

© 2026 Magimo

Sheet	Primary purpose	Typical editor
“Validation”	Test cases, evidence references, result status and release coverage	Creator or quality owner
“ReleaseInfo”	Model identification, version, revision, compatibility and release state	Release owner
“DocsIndex”	References to external documents, manuals, reports and records	Creator or release owner

Not every model requires every sheet. A simple, single-user domestic part may need only clearly separated input, engineering and status areas. Separate sheets become increasingly valuable when a model is shared, maintained by others, connected to automation, distributed as a configurable product or used where failure has more significant consequences.

B.3 Common Spreadsheet conventions

B.3.1 General

Apply the same conventions across all sheets.

- Use stable English aliases without spaces, punctuation or cell-location references.
- Use explicit units for every physical quantity.
- Keep the human-readable label separate from the alias.
- Place the value in one clearly identified cell per parameter.
- Include a short description where a parameter could be misunderstood.
- Use a consistent status vocabulary, such as Draft, Experimental, Validated, Supported, Released and Deprecated (§6.1).
- Avoid references to raw cell addresses in sketches, expressions, macros and external interfaces.
- Do not duplicate an independent engineering value in multiple sheets.

A practical row layout is:

Table B–2: Row layout of a parameter sheet

Field	Purpose
Parameter ID	Stable identifier, for example ‘wallThickness’
Display name	Readable label, for example “Wall thickness”
Value	Current value or formula
Unit	Unit where applicable
Type	Context, choice, engineering, derived, diagnostic or metadata
Owner	User, creator, manufacturing owner or release owner
Status	Draft, supported, validated, etc.
Description	Short explanation of meaning and intended use
Evidence or source	Test reference, standard, design rationale or profile reference

The exact column order may vary, but the meaning should remain consistent throughout the model.

B.3.2 Colour scheme

Assign distinct colours to user inputs, approved choices, creator-controlled engineering values, physical or material constants and calculated outputs. Mark geometry-driving values clearly and present driven or reference values in a visually subordinate style. Colour supports the parameter contract; it should not be the only means of distinguishing an editable input from a creator-controlled or derived value.

Table B–3: Colour scheme

Cell function	Suggested fill	Code (HTML)	Code (RGB)	Rule
User input / context value	Blue	#99C6FF	153, 198, 255	Boundaries (e.g. printer envelope), editable by the intended user
User choice / configuration selector	Light blue	#D1E8FF	209, 232, 255	Discrete approved option, driving parameter
Creator-controlled engineering value	Light green	#CCF2CC	204, 242, 204	Visible but creator-controlled; includes design rules, margins and geometry-driving values
Physical or material constant	Grey	#DBDBDB	219, 219, 219	Fixed reference data; do not edit
Calculated / derived value	White, set explicitly	#FFFFFF	255, 255, 255	Read-only result. Can be input for other calculations
Driving parameter	(Bold text)	–	–	Controls downstream geometry
Driven / reference value	Grey italic text	Text #808080	128, 128, 128	Reports a geometric result; not an input
Validation status / diagnostic	Amber	#FFDB8C	255, 219, 140	The status text (OK, Review, Reject) shows the state; FreeCAD cannot colour a cell by its value
Header row	Light grey	#EDEDED	237, 237, 237	Column headings

Notes:

- FreeCAD Spreadsheet cells do not provide robust cell-level locking or access control. Use separate sheets, release copies, file-management controls and documented editing responsibilities to reduce the probability of accidental changes.
- FreeCAD fills an aliased cell that has no colour of its own with a preference colour. The default is a bright yellow, but themes may change it: OpenLight, for example, uses #BAC8FF, a blue close to the user-input colour. Give every aliased cell an explicit fill, derived values included, so the scheme looks the same under every theme. The preference is AliasedCellBackgroundColor, under Tools → Edit parameters → BaseApp → Preferences → Mod → Spreadsheet.

© 2026 Magimo

- Record the colours as codes, not only as names such as "blue" or "grey". The exact appearance still varies with the operating system, theme and screen. Colour therefore supports the parameter specification but never replaces it.
- FreeCAD 1.1.2 offers a grid of preset colours in the toolbar and in the Cell Properties dialog. The inconspicuous "···" button at the bottom of the grid opens a full colour dialog: enter the HTML code there. A colour entered this way stays available in the grid as a custom colour.

B.4 UIParams: the user configuration sheet

“UIParams” is the public configuration interface inside the FreeCAD file. It should contain only values that the intended user, customer, manufacturing operator or external configurator is genuinely expected to provide or select. Typical entries include:

- Mating dimensions.
- Available installation space.
- Required overall length.
- Approved fit class.
- Mounting option.
- Selected material profile.
- Approved nozzle or process profile.
- A product variant or feature option.
- A user-controlled orientation choice, but only where more than one orientation is supported and the choice does not bypass a required manufacturing or structural condition.

Table B–4: UIParams structure (example)

Parameter ID	Display name	Value	Unit	Input type	Supported range or options	Description
matingDiameter	Mating diameter	20	mm	Numeric	15 mm to 40 mm	Diameter of the mating component
installationLength	Installation length	80	mm	Numeric	50 mm to 150 mm	Required functional length
fitClass	Fit class	“Normal”	–	Choice	Loose, Normal, Tight	Approved interface fit category
materialProfile	Material profile	‘PETG_STD_01’	–	Choice	Approved profiles only	Material and associated validated rules
mountingVariant	Mounting variant	‘ScrewMount’	–	Choice	ScrewMount, ClipMount	Selected mounting concept

Do not use “UIParams” to expose raw design constants, shrinkage factors, wall formulas, safety factors, feature offsets or internal fillet radii. Those values belong in “CMParams” or a controlled profile sheet.

© 2026 Magimo

> Where a user value needs checking, retain the requested value as the input and calculate a separate resolved, derived or validated value elsewhere. Do not silently replace an entered value without making the behaviour and resulting value visible.

B.5 CMParams: the creator-control sheet

"CMParams" contains the creator-controlled engineering implementation. It is the preferred place for calculations that translate user inputs and process data into geometry-driving values. Typical entries include:

- Minimum wall rules.
- Derived clearance values.
- Shrinkage compensation.
- Internal transition lengths.
- Fillet radii.
- Rib and gusset rules.
- Feature-spacing calculations.
- Build-envelope checks.
- Intermediate geometric values.
- Diagnostic flags and messages.

Table B–5: CMParams structure (example)

Parameter ID	Display name	Value or expression	Unit	Type	Description
minWallRequired	Required minimum wall	Formula based on active process profile	mm	Engineering	Minimum wall required by the approved process profile
fitClearance	Resolved fit clearance	Formula based on fit class and material profile	mm	Derived	Clearance resolved through controlled rules
outerDiameter	Outer diameter	UIParams.matingDiameter + 2 * fitClearance + 2 * wallThickness	mm	Derived	Geometry-driving outer interface diameter
wallStatus	Wall status	Diagnostic expression	–	Diagnostic	Indicates whether wall requirements are met
buildStatus	Build-volume status	Diagnostic expression	–	Diagnostic	Indicates whether the configuration fits the permitted envelope

Use CMParams to keep expressions manageable. Rather than placing long compound formulas directly in many sketches, calculate the meaningful intermediate value once and reference that alias from the geometry. A useful pattern is: user input or approved choice → creator-controlled rule → derived geometry value → geometry and diagnostic status. This makes the dependency chain easier to inspect, test and maintain.

© 2026 Magimo

B.6 MaterialProfile and ProcessAspects

Keep material data and process assumptions distinguishable even when they are closely related.

- "MaterialProfile" records controlled material-specific information, such as:
  - Material identifier and revision.
  - Material family.
  - Density.
  - Modulus or stiffness category where relevant.
  - Shrinkage or dimensional compensation values.
  - Moisture sensitivity or conditioning requirements.
  - Permitted process window.
  - Material qualification status.
  - Known exclusions or limitations.
- "ProcessAspects" records the manufacturing context and manufacturing obligations, such as:
  - Printer or printer class.
  - Nozzle diameter.
  - Layer-height range.
  - Build-volume limits.
  - Required print orientation.
  - Support restrictions.
  - Minimum wall and feature rules.
  - Bridge or overhang rules.
  - Required shell or perimeter assumptions.
  - Post-processing requirements.

For each process aspect, identify whether it affects geometry, function, safety or guidance only. This distinction prevents a recommendation from being mistaken for a controlled design rule.

Table B–6: ProcessAspects sheet (example)

Aspect ID	Value	Effect	Status	Enforcement method	User-facing message
nozzleDiameter	0.4 mm	Geometry	Required	Model and profile	Use the specified nozzle profile
printOrientation	Face A down	Function	Required	Documentation and validation	Place marked face on build plate
infillDensity	40%	Function	Minimum requirement	Manufacturing manifest	Do not reduce below 40%
seamPreference	Rear face	Guidance	Recommended	Documentation	Keep seam away from visible front face

© 2026 Magimo

B.7 UseConstraints and Validation

“UseConstraints” contains concise, structured facts about the intended application. It should support, rather than replace, a separate use and safety document. Relevant fields may include:

- Intended function.
- Prohibited use.
- Maximum static and dynamic loads.
- Permitted load direction.
- Maximum lever arm or load position.
- Service-temperature conditions of use.
- Chemical or environmental exclusions.
- Inspection requirement.
- Retirement criteria.
- Foreseeable misuse.
- Required assembly orientation.

“Validation” records evidence that connects a configuration to an actual level of confidence. Each record should identify the tested configuration, model version, material profile, process profile, build orientation, result, status and supporting evidence.

Table B–7: Validation record (example)

Test ID	Config. ID	Material profile	Process profile	Build Orientation	Result	Status	Evidence reference
VT-001	CFG-DEFAULT	PETG_STD_01	FDM_04_020	Face A down	Passed dimensional check	Supported	TR-001
VT-002	CFG-MAX-LENGTH	PETG_STD_01	FDM_04_020	Face A down	Passed functional test	Validated	TR-002
VT-003	CFG-TIGHT-FIT	PA12_STD_01	FDM_06_020	Face A down	Marginal assembly force	Experimental	TR-003

A configuration that recomputes successfully is not automatically validated. The “Validation” sheet should make that distinction visible.

B.8 ReleaseInfo and DocsIndex

“ReleaseInfo” identifies the released combination of the model, applicable profiles and controlled documentation. Minimum fields include:

- Model identifier.
- Model version.
- Release date.
- Release status.

- Compatible material-profile versions.
- Compatible process-profile versions.
- Supported configuration-domain statement.
- Known limitations.
- Revision summary.
- Release approver or responsible role.

“DocsIndex” links the FreeCAD model to controlled external documentation. Use document identifiers and revision numbers rather than relying only on informal filenames.

Document type	Document ID	Version	Status	Location or reference
Manufacturing instructions	MI-001	1.0	Released	Relative path or controlled URL
Use and safety instructions	US-001	1.0	Released	Relative path or controlled URL
Validation report	TR-002	1.0	Released	Relative path or controlled URL
Material profile guide	MP-PETG-001	1.2	Released	Relative path or controlled URL

Keep the model file self-identifying. Even if external documents are unavailable, the `.FCStd` file should identify the model version, release status, key limitations and associated document identifiers.

B.9 Practical implementation techniques

Use the following practices to make the Spreadsheet architecture easier to operate and review.

- Tile the 3D view and Spreadsheet windows during parameter development so that geometry changes remain visible.
- Use aliases in the Expression editor rather than raw cell addresses.
- Apply the documented colour convention consistently; do not introduce a second colour meaning elsewhere in the model.
- Use a separate “Debug” Body or group for temporary geometry, test sketches and diagnostics.
- Reduce the risk of accidental changes through separate sheets, visual conventions, release copies, controlled access to source files and documented editing responsibilities. Do not represent these measures as security control.
- Include a small number of readable diagnostic cells, such as `wallStatus`, `fitStatus`, `buildStatus` and `configurationStatus`.
- Use a release copy of the file and remove or isolate temporary material before distribution.
- Export the parameter schema to CSV, JSON or another controlled format when the model will be connected to a platform, script or external configurator.

Keep diagnostics understandable. A warning should state what failed, why it matters and what action is required. For example, “FAIL: required wall is 2.4 mm; calculated wall is 1.8 mm” is more useful than a generic error flag.

© 2026 Magimo

B.10 Final implementation check

Before connecting the Spreadsheet architecture fully to geometry, review the model using the following questions:

- Can a new maintainer identify the intended user-facing inputs without inspecting sketches?
- Does every independent engineering value have one authoritative definition?
- Are user inputs, creator-controlled values and derived values visibly separated?
- Are units, aliases, descriptions, defaults and permitted ranges clear?
- Are material and process assumptions identified and versioned?
- Does the geometry consume controlled or derived values where appropriate?
- Are diagnostic checks visible for important geometric, manufacturing and functional limits?
- Can the released model be associated with validation evidence and external documentation?
- Does the file make its version, release status, supported domain and known limitations clear?

If these questions cannot be answered from the Spreadsheet layer and release information, the model may still be operable, but it may not yet be easy to hand over, review or maintain. Apply the checklist proportionately to the intended use: not every simple model needs every record, but a released or shared model should make its engineering contract inspectable before a user has to interpret the geometry itself.

© 2026 Magimo

Appendix C Configuration Tables in FreeCAD

C.1 Purpose

A configuration table is a useful tool for managing a limited set of approved design variants. It is not required for every FreeCAD model and should not replace direct parametric relationships where those are clearer. Use a configuration table when a model must translate a limited set of recognised selections or combinations into a defined set of engineering values, geometry rules, manufacturing requirements or release states.

The table acts as a controlled lookup structure between a meaningful configuration choice and the model response. For example, a user may select:

- A fit class: “Loose”, “Normal” or “Tight”
- A material profile: `PETG_STD_01`, `PA12_CF_01` or `TPU_95A_01`
- A product size: “Small”, “Medium” or “Large”
- A standard hardware interface: “M3”, “M4” or “Aluminium profile, such as 20x20 or 30x40”
- A reinforcement level: “Light”, “Standard” or “Heavy”

The table can return associated controlled values such as clearance, minimum wall thickness, hole compensation, edge treatment, permitted orientation, reinforcement geometry or validation status. A configuration table is therefore not an arbitrary list of dimensions. It is a controlled engineering lookup. It makes a defined part of the supported design domain visible and helps prevent a user-facing choice from becoming an uncontrolled edit to internal geometry.

> **A configuration table is a controlled lookup, not a user interface**
>
> A configuration table can return coordinated engineering values for a limited set of approved choices. It does not by itself provide a polished selector, filter incompatible combinations or validate the resulting geometry. Keep the user-facing interface, lookup logic, engineering checks and release status distinct, and ensure that unsupported keys or combinations fail visibly.

This appendix explains when configuration tables are useful, how they can be organised in FreeCAD Spreadsheets and how to connect them safely to a parametric model. It supplements the parameter architecture, manufacturing-profile, validation and release principles explained in the main chapters.

C.2 When to use one

Use a configuration table where the number of meaningful variants is limited and every variant can be defined, reviewed, maintained and, where necessary, validated by the creator. A table is particularly useful when:

- A single user choice changes several engineering values together.
- The permitted variants are discrete rather than continuous.
- Each variant has a clear functional or manufacturing meaning.
- The relevant combinations can be tested or otherwise justified.
- The selected option must return several coordinated values.

© 2026 Magimo

- The model must display a release, validation or support status for the selected configuration.
- A standard component, profile or product variant has known dimensions and rules.

Typical applications include:

- Fit classes that map to clearance, wall rule and assembly method.
- Material and nozzle combinations that map to minimum feature size, wall thickness and shrinkage compensation.
- Standard hardware sizes that map to pocket dimensions, lead-in geometry and required radial wall.
- Product variants that map to defined geometry and intended-use limits.
- Reinforcement levels that map to rib count, rib thickness, shell requirement and weight target.
- Closed manufacturing environments with a limited set of approved machines or process profiles.

A table is suitable where there is a limited set of standard choices. It is not the right tool for a physical input that is genuinely continuous. A cable diameter, mounting-hole pitch, hand width, available installation length or printer build volume is generally better entered directly as a measured value and processed through formulas.

The practical rule is:

> Use a configuration table when the creator can define and justify every row. Use direct parameters and formulas when the physical situation is continuous.

C.3 What it does not replace

A configuration table complements, rather than replaces, the ordinary parametric tools available in FreeCAD.

Table C–1: Overview of parametric tools

Tool	Best used for	Example
Direct parameter	A measured or required continuous value	Cable diameter, required length, mating diameter
Expression	A consistent mathematical relationship	Outer diameter derived from mating diameter, clearance and wall thickness
Sketch constraint	A geometric relationship belonging in the sketch	Symmetry, tangency, equal spacing, alignment
Configuration table	A limited set of approved alternatives	Fit class, standard washer size or material-process profile
Script or external logic	Large datasets, complex optimisation or automated generation	Product catalogue, high-count pattern generation, platform-driven batch output

Do not create a configuration table merely because a formula appears complex. If the underlying relationship is continuous and understandable, a named expression in “CMParams” is normally clearer and more scalable.

© 2026 Magimo

Conversely, do not force a continuous formula onto a decision that is inherently categorical. A fit class, mounting concept, edge treatment or reinforcement pattern often represents a deliberately bounded design choice. A configuration table can describe such choices more clearly than a long chain of conditional expressions distributed throughout the model.

C.4 Basic concepts

A configuration table contains three essential elements:

1. Configuration Key – the identifier that selects a recognised configuration.
2. Selector – the input or combination of inputs used to determine the key.
3. Result values – the controlled values returned by the selected configuration.

These elements are discussed in the following sections.

C.4.1 Configuration Key

The Configuration Key is the stable identifier used by the model to select a defined row or column in a table. Examples include:

- FIT_LOOSE
- PA12CF_STD_01
- M6_STD
- HEAVY_DUTY

A Configuration Key is an identifier, not a dimension.

> **Numeric-looking keys remain text**
>
> FreeCAD treats a Configuration Key as text. A key such as 04, 1 or 100 may therefore be accepted, but it is a text string, not a number. This can create confusion later: 04 may look like a 0.4 mm nozzle size, while 100 may look like a hole count or a dimension.
>
> Use text-based names that remain meaningful when the table is reviewed, exported or connected to an external configuration interface. Prefer keys that state their meaning explicitly, such as NOZZLE_04, HOLES_1 or SIZE_100. If a numeric value is genuinely needed by the model, store it separately as a value with an explicit unit and derive or validate it through the normal parameter logic. Do not use a Configuration Key as a substitute for a numeric parameter.
>
> If a number is used to identify a variant, format it so that it cannot be mistaken for a numeric value. For example, use a trailing underscore, such as 1_.

The key should normally originate in “UIParams”, where it represents a genuine user choice, or in “CMParams”, where it is calculated from validated user inputs. The geometry should not use the key directly. It should consume the controlled result values selected by that key.

C.4.2 Selector

A selector is the input used to identify the required table row or column. It may be a single choice or a controlled combination of choices.

© 2026 Magimo

Table C–2: Simple selector (example)

User input	Configuration Key
Normal fit	FIT_NORMAL
PA12-CF	PA12CF_STD_01
M5	M5_STD

A compound selector may combine more than one controlled input:

Table C–3: Compound selector (example)

Material profile	Nozzle size	Configuration Key
PETG_STD_01	0.4 mm	PETG_04
PETG_STD_01	0.6 mm	PETG_06
PA12CF_STD_01	0.4 mm	PA12CF_04

Use compound selectors only where the combination represents a real engineering dependency. Do not create combinations merely because several independent parameters happen to exist.

C.4.3 Result values

Result values are the controlled outputs associated with the selected configuration. They may drive geometry, validation rules, manufacturing guidance or release information. Examples include:

- Clearance allowance
- Minimum wall thickness
- Minimum radial wall around a hole
- Hole or pocket compensation
- Shrinkage compensation
- Preferred print orientation
- Permitted support strategy
- Required number of perimeters
- Maximum supported load
- Edge treatment
- Validation status
- Documentation reference

Each result value must have one authoritative definition. If a clearance value exists in a configuration table, a sketch constraint and a separate engineering Spreadsheet cell, the model contains competing sources of truth.

C.5 Table orientation

A configuration table can be arranged horizontally or vertically. Choose the arrangement that makes the engineering meaning easiest to inspect and maintain.

C.5.1 Horizontal table

In a horizontal table, each row represents one complete configuration. The first column normally contains the Configuration Key and the following columns contain the result values.

Table C–4: Horizontal table layout (example)

Configuration Key	Fit class	Clearance	Minimum wall	Assembly method	Status
FIT_LOOSE	Loose	0.40 mm	2.0 mm	Hand assembly	Validated
FIT_NORMAL	Normal	0.25 mm	2.4 mm	Functional sliding fit	Validated
FIT_TIGHT	Tight	0.10 mm	2.8 mm	Light press fit	Experimental

The horizontal layout is usually best when:

- Each configuration has several result values.
- Each row represents a complete engineering statement.
- New configurations may be added over time.
- The table will be exported to CSV, JSON or an external configuration system.
- The creator needs to review one configuration at a time.

For most engineering configuration tables, a horizontal row-based layout is the preferred default.

C.5.2 Vertical table

In a vertical table, each column represents one configuration and each row represents a property.

Table C–5: Vertical table layout (example)

Parameter	Loose	Normal	Tight
Clearance	0.40 mm	0.25 mm	0.10 mm
Minimum wall	2.0 mm	2.4 mm	2.8 mm
Assembly method	Hand	Functional	Press
Validation status	Validated	Validated	Experimental

This layout is useful when:

- There are only a few configurations.
- The main purpose is comparison between variants.
- The table will be reviewed visually within FreeCAD.
- Each configuration has a modest number of outputs.

A vertical table becomes difficult to maintain when many configurations are added. It can also become inconvenient when the same data must be exported or interpreted by an external system.

C.6 Table layout and aliases

A configuration table should be readable by people and stable for expressions. Use:

- A clear table title.

- Human-readable column or row labels.
- Stable aliases for every value used by expressions.
- Units in cells containing physical values.
- A description or note column where the engineering meaning is not obvious.
- A status field where validation or support differs by configuration.
- A version or revision identifier for the table where the model is distributed.

For a horizontal table, a useful structure is:

Table C–6: Structure of a horizontal table (example)

Key	Display name	Result 1	Result 2	Result 3	Status	Note
FIT_LOOSE	Loose fit	0.40 mm	2.0 mm	Hand assembly	Validated	Easy manual assembly
FIT_NORMAL	Normal fit	0.25 mm	2.4 mm	Functional sliding fit	Validated	Default option
FIT_TIGHT	Tight fit	0.10 mm	2.8 mm	Light press fit	Experimental	Verify on target process

For a vertical table, use a human-readable header row and a deliberate alias structure. Do not allow geometry to depend on unexplained cell addresses such as “C14” or “D18”. The alias is part of the model’s engineering vocabulary. Use names such as:

- normalClearance
- normalMinWall
- m5HoleDiameter
- m5NutTrapWidth
- tpuMinThickness
- petgShrinkageXY

Avoid aliases such as:

- value1
- data2
- cellC14
- tempValue
- newClearance

An alias should describe what the value means, not where it happens to be stored.

C.7 Retrieving values in FreeCAD

A configuration table is useful only when the selected configuration reliably supplies the values used by the model. In FreeCAD, the lookup can be implemented in several ways. Choose the simplest method that makes the data flow clear and that you can test in the target FreeCAD version. Three approaches are common:

- Native Configuration Table.
  Use FreeCAD's Configuration Table command to associate predefined Spreadsheet values with a configuration property on an object.
- Centralised expressions.
  Use a key and a controlled expression or mapping layer to select the required values.
- Scripted or external lookup.
  Resolve larger or more complex tables in a script, database or external configurator, then pass controlled outputs to the FreeCAD model.

For a small number of discrete variants, the native Configuration Table or a short, centralised expression is usually easiest to inspect. For larger catalogues or more complex compatibility rules, a controlled external lookup may be more suitable. In every case, geometry should consume the selected engineering values rather than duplicate the lookup logic throughout sketches and features.

C.7.1 Native FreeCAD Configuration Tables

FreeCAD's Spreadsheet workbench includes a Configuration Table command. It associates a named configuration with a set of predefined values. The selected configuration is exposed as a property on the object to which the table is bound. The property can then be changed in the object's Data properties. The Spreadsheet cells themselves do not provide the selector control. A typical table has:

- A header row containing the parameter names.
- A row reserved for the currently selected configuration.
- One or more configuration rows containing named variants and their values.
- Aliases on the current-value cells that the model needs to reference.

For example, a fit table might contain values for clearance and minimum wall thickness:

Table C–7: Fit table (example)

Configuration	Clearance	Minimum wall thickness
FITLOOSE	0.40 mm	2.0 mm
FITNORMAL	0.25 mm	2.4 mm
FITTIGHT	0.10 mm	2.8 mm

The cells holding the current clearance and minimum-wall values should have meaningful aliases, such as `fitClearance` and `minWallRequired`. Expressions in the model should reference those aliases, not cells in the individual configuration rows. The selected configuration then changes the values available through the current-value aliases, while the geometry keeps a consistent set of references.

The precise range and layout of a Configuration Table depend on its setup. Follow the structure shown in the Configuration Table dialog and verify the resulting object property and Spreadsheet values in a small test document before applying the pattern to a released model. Do not assume that a layout copied from another project, or a vertical arrangement, is interchangeable without testing it. A practical setup and verification sequence is:

1	Create a small table with a header, a current-configuration row and at least two named variants.

2	Use Configuration Table → Setup Configuration Table to define the table and configuration property.
3	Assign clear aliases to the current-value cells that will drive the model.
4	Bind geometry-driving expressions to those aliases.
5	Change the configuration property and confirm that the current values and dependent geometry update as intended.
6	Save, close and reopen the document, then repeat the check to confirm that the configuration is preserved.

FreeCAD exposes the selector through the bound object's Data properties. It is not a drop-down embedded in a Spreadsheet cell. If the intended user needs a more guided interface, provide a separate dialog, task panel or external configuration form. Such an interface should set the same controlled configuration property or pass values through the same documented parameter contract. It should not create a parallel source of engineering values.

C.7.2 Direct alias references

Direct alias references are appropriate when the model uses a known configuration or when a creator is comparing or qualifying individual variants. Store each controlled result under a meaningful alias, then reference the required alias explicitly. For example, a fit table may define:

Table C–8: Fit values stored under aliases

Alias	Value
looseClearance	0.40 mm
normalClearance	0.25 mm
tightClearance	0.10 mm
looseMinWall	2.0 mm
normalMinWall	2.4 mm
tightMinWall	2.8 mm

A model expression can then use the selected alias, for example ConfigTable.normalClearance. This method is simple and readable, but the model or expression must be changed when a different configuration is required. It is best suited to a fixed configuration, a small test model or a workflow in which the creator, rather than the end user, chooses the variant. Do not use direct references to variant cells as a substitute for a clear user-facing selector.

C.7.3 Centralised conditional expressions

A centralised expression can map a recognised Configuration Key to the corresponding table values. The general logic is:

1	The user selects a permitted key, such as FITNORMAL.
2	The model checks that the key is recognised.
3	A single controlled lookup layer resolves the key to result values.
4	Geometry consumes those resolved values.
5	An unknown or unsupported key produces a clear diagnostic or controlled rejection.

Keep this mapping in one location. Do not repeat a separate chain of conditions in each sketch, feature or Body. If adding a variant requires changing the selection logic in several places, consolidate the lookup before extending the table.

Expression syntax and available functions can vary by FreeCAD version and expression context. Treat any expression used in a project as illustrative until it has been tested in the target version. Where the mapping becomes long or difficult to audit, use a native Configuration Table or move the lookup to a controlled script or external system instead.

C.7.4 Scripted or external lookup

For catalogue-scale data, many result fields or complex compatibility rules, a script or external configurator may be easier to maintain than a large collection of Spreadsheet expressions. It can validate a selection, resolve a Configuration Key, retrieve controlled results, check profile compatibility and pass the resulting values to FreeCAD. This approach adds another controlled component to the product. Record and version the lookup data and logic alongside the model. Validate inputs before generating geometry, preserve the selected configuration and relevant profile versions, and ensure that the external system applies the same hard rules as the model. It must not silently replace a requested configuration or bypass a rejection rule.

A practical division of responsibility is:

- The user-facing interface captures the permitted choices and context.
- The lookup layer maps those inputs to controlled engineering values.
- The FreeCAD model uses the resolved values to generate and check geometry.
- The validation and release records identify the configuration, model and profile versions, and evidence.

C.7.5 Independent selectors and dependent choices

Multiple Configuration Table properties can provide several selectors, for example a product variant, a mounting option and a material profile. These are independent selectors unless additional logic constrains how they may be combined. Their presence does not, by itself, create a parent–child selector hierarchy.

If some combinations are incompatible, represent and check that compatibility explicitly. For example, a selected product variant may permit only certain mounting options. The model or its configuration interface should reject or flag an unsupported combination before release or manufacture.

A genuinely dependent interface, where a parent choice filters the options available in a second selector, requires additional interface or scripted logic. Keep that filtering logic separate from the engineering lookup and validate the final combination in the model or its controlled generation boundary. Do not imply that multiple native selectors automatically filter one another. Use multiple selectors only when each represents a meaningful choice and the permitted combinations can be defined and tested. Where a single Configuration Key can represent the complete approved combination more clearly, prefer that simpler interface.

C.7.6 Selection and lookup checks

Whichever retrieval method is used, test the complete path from the user's selection to the released result. At minimum, verify that:

- Every supported key resolves to the intended result values.
- The default selection produces the intended default geometry.

© 2026 Magimo

- Changing the selection updates every dependent value and feature.
- Aliases point to the current or resolved values – not to a fixed variant row.
- Unknown keys and incompatible combinations are rejected or clearly flagged.
- The active geometry and exported result correspond to the selected configuration.
- Warnings and validation status update with the configuration.
- The selected key and relevant table, model and profile versions can be recorded for traceability.

A successful recompute shows that FreeCAD generated a result. It does not establish that the result is printable, functionally suitable or supported. Assess the selected configuration against the applicable validation and release requirements in Chapters 6 and 7 and the relevant appendix sections.

C.8 Recommended architecture

A practical arrangement is:

- "UIParams" contains the user-facing selection, for example `fitClass`, `materialProfile` or `washerSize`.
- "ConfigTable" contains approved Configuration Keys and controlled result values.
- "CMParams" resolves selected results into geometry-driving values and diagnostic statuses.
- sketches and features reference derived aliases in "CMParams".
- "Validation" records the status and evidence associated with configurations or profiles.

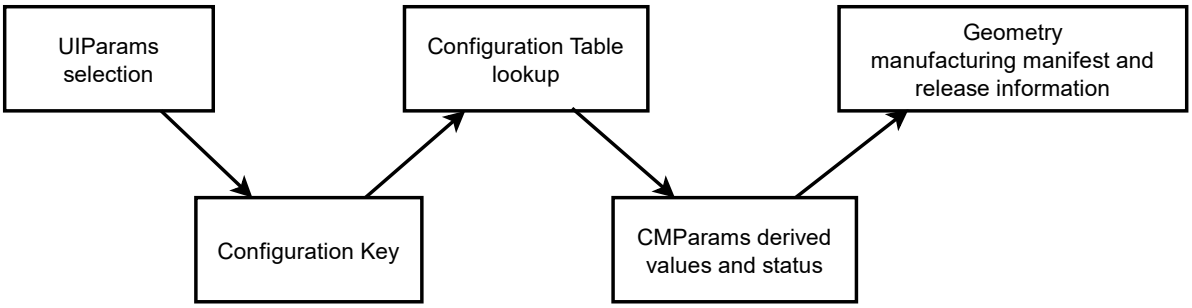


Figure C–1: The information flow

Example

Layer	Alias	Example value
UIParams	fitClass	FIT_NORMAL
ConfigTable	normalClearance	0.25 mm
ConfigTable	normalMinWall	2.4 mm
CMParams	fitClearance	Selected clearance
CMParams	minWallRequired	Selected wall rule
CMParams	fitStatus	"VALIDATED"
Sketch or feature	Constraint expression	"CMParams.fitClearance"

This keeps configuration selection, engineering implementation and geometry separate.

C.9 Worked example: parametric pen holder

This example demonstrates a configuration table used to generate a cuboid desk pen holder containing one, two or three holes of different diameters. The product may contain:

- Pen holes: 10 mm diameter.
- Ballpoint holes: 16 mm diameter.
- Marker holes: 20 mm diameter.

The user selects how many holes of each type are required. The model determines the valid hole sequence, calculates the product outer dimensions and positions each hole individually. The example is useful because the holes have different diameters. A simple linear pattern cannot be applied reliably because hole positions depend on the selected hole sequence.

Note that this is a simplified demonstrator for the use of a configuration table, not a fully engineered or validated product. It omits checks and features that a released pen holder would require, including fillets, chamfers, tapered or relieved pocket holes, material-balance considerations, stress-relief features, manufacturing allowances and detailed validation. The screenshots use an earlier colour coding than §B.3.2; the colours differ, but the cell roles are the same.

C.9.1 User inputs

Table C–9: Pen holder: user inputs

Input	Meaning	Allowed values
nPen	Number of 10 mm pen holes	0 to 3
nBallpoint	Number of 16 mm ballpoint holes	0 to 3
nMarker	Number of 20 mm marker holes	0 to 3

The total number of holes must be between one and three:

$$0 < (nPen + nBallpoint + nMarker) \leq 3$$

The configuration `000` is invalid because it produces no holes. A configuration such as `400` is invalid because it exceeds the permitted maximum.

C.9.2 Configuration Key

The model can derive a Configuration Key from the three user counts:

```
Configuration Key = nPen * 100 + nBallpoint * 10 + nMarker
```

Table C–10: Pen holder: user selections and Configuration Keys

User selection	Configuration Key
1 pen, 0 ballpoints, 0 markers	100
0 pens, 1 ballpoint, 0 markers	010
2 pens, 1 ballpoint, 0 markers	210
1 pen, 1 ballpoint, 1 marker	111

© 2026 Magimo

The user does not need to enter the key directly. The model derives it from controlled numerical inputs after confirming that the counts are valid. The process is:

1. The user enters the required counts.
2. The model checks the individual and total limits.
3. The model generates the Configuration Key.
4. The key selects a recognised configuration.
5. The configuration table returns the appropriate hole sequence and values.
6. The geometry consumes those controlled results.

C.9.3 Geometry rules

Let `thumbWidth` be the creator-controlled ergonomic margin used both between the outer faces and the adjacent hole edges, and between adjacent holes. It provides the finger access required to grasp and remove a pen, ballpoint or marker.

- `thumb_width` = creator-controlled edge and finger-access margin
- `d_i` = diameter of hole i; inactive holes have d_i = 0
- `nTot` = nPen + nBallpoint + nMarker

The total product length is the sum of the active hole diameters, plus `thumb_width` at each outer edge and between each pair of adjacent holes:

$$product_length = d_1 + d_2 + d_3 + (nTot + 1) * thumb_width$$

The product width is the largest active hole diameter plus `thumb_width` at each side. Hole centres are calculated sequentially from the selected hole sequence; inactive holes have zero diameter and are suppressed:

$$product_width = 2 * thumb_width + max(d_1; d_2; d_3)$$

The first hole centre is:

$$x_1 = thumb_width + \frac{d_1}{2}$$

Each following active hole centre is:

$$x_i = x_{i-1} + \frac{d_{i-1}}{2} + thumb_width + \frac{d_i}{2}$$

Inactive holes have zero diameter and zero position; their features are suppressed.

C.9.4 Calculated values

Add these aliases to the parameter Spreadsheet:

Table C–11: Pen holder: calculated values

Alias	Meaning
nPen	Number of 10 mm pen holes
nBallpoint	Number of 16 mm ballpoint holes
nMarker	Number of 20 mm marker holes
thumb_width	Edge and finger-access margin
nTot	Total number of active holes

Alias	Meaning
config_key	Derived configuration key
d_1, d_2, d_3	Hole diameters in placement order
x_1, x_2, x_3	Hole-centre positions
product_length	Outer length
product_width	Outer width
config_status	Validation result

C.9.5 Counts and key

```
nTot = nPen + nBallpoint + nMarker

config_key = nPen * 100 + nBallpoint * 10 + nMarker

config_key_text = str(nPen) + str(nBallpoint) + str(nMarker) + <<_>>
```

The numeric expression nPen * 100 + nBallpoint * 10 + nMarker is used only to derive and validate the selected combination. The Configuration Key itself is stored as text and is given a trailing underscore, for example 210_. This makes clear that the key is an identifier, not a numerical dimension or calculation result.

C.9.6 Hole diameters

Because the placement order is pens, ballpoints, then markers, the diameters can be derived directly from the counts in FreeCAD:

```
d_1 = nPen > 0 ? 10mm : (nBallpoint > 0 ? 16mm : 20mm)

d_2 = nTot < 2 ? 0mm : (nPen > 1 ? 10mm : (nBallpoint > 0 ? 16mm :
20mm))

d_3 = nTot < 3 ? 0mm : (nPen > 2 ? 10mm : (nBallpoint > 1 ? 16mm :
20mm))
```

These formulas reproduce every valid sequence in the configuration table.

C.9.7 Hole positions

```
x_1 = thumb_width + d_1 / 2

x_2 = nTot < 2 ? 0mm : (x_1 + d_1 / 2 + thumb_width + d_2 / 2)

x_3 = nTot < 3 ? 0mm : (x_2 + d_2 / 2 + thumb_width + d_3 / 2)
```

C.9.8 Product envelope

```
product_length = d_1 + d_2 + d_3 + (nTot + 1) * thumb_width
```

For the width, use nested conditionals rather than relying on a three-argument `max()`:

```
d_max = d_1 > d_2 ? (d_1 > d_3 ? d_1 : d_3) : (d_2 > d_3 ? d_2 : d_3)

product_width = 2 * thumb_width + d_max
```

C.9.9 Validation status

```
config_status = and(nPen >= 0; nPen <= 3;

   nBallpoint >= 0; nBallpoint <= 3;

   nMarker >= 0; nMarker <= 3;

   nTot >= 1; nTot <= 3)

? <<OK>> : <<INVALID>>
```

For stricter validation, add a check that the derived key exists in the configuration table. In a spreadsheet implementation, this can be a separate `key_valid` lookup or a nested conditional covering the 19 permitted keys; the geometry should remain suppressed unless:

```
config_status == <<OK>>
```

C.9.10 Worked example: configuration 111

With `nPen` = 1, `nBallpoint` = 1, `nMarker` = 1 and `thumb_width` = 25 mm, the following results are obtained:

Table C–12: Pen holder: results for configuration 111

Value	Formula result
nTot	3
config_key	111
d_1	10 mm
d_2	16 mm
d_3	20 mm
x_1	30 mm
x_2	68 mm
x_3	111 mm
product_length	146 mm
product_width	70 mm
config_status	OK

The configuration table remains useful as the controlled record of permitted keys, hole sequences and validation status, while the formulas generate the dependent geometry. The hole sequence and diameters determine the dependent positions. Hole positions should therefore not be entered manually as independent values.

C.9.11 Configuration table

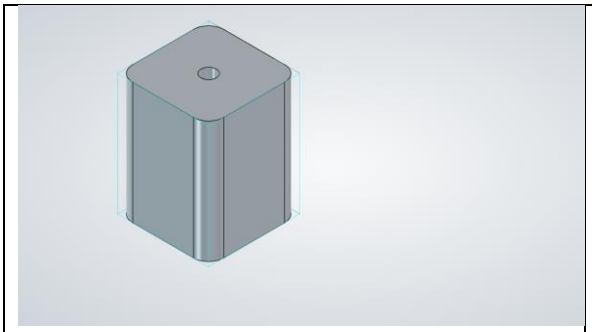
The configuration table defines each valid combination containing one, two or three holes. Note that `Key` uses the `config_key_text`, as defined above, to ensure FreeCAD treats the Key as a text string. The trailing underscore also prevents a leading zero from being omitted.

Table C–13: Pen holder: Configuration Table

Key	Pens	Ballpoints	Markers	Hole sequence	Total holes
100_	1	0	0	10	1
010_	0	1	0	16	1
001_	0	0	1	20	1
200_	2	0	0	10, 10	2
020_	0	2	0	16, 16	2
002_	0	0	2	20, 20	2
110_	1	1	0	10, 16	2
101_	1	0	1	10, 20	2
011_	0	1	1	16, 20	2
300_	3	0	0	10, 10, 10	3
030_	0	3	0	16, 16, 16	3
003_	0	0	3	20, 20, 20	3
210_	2	1	0	10, 10, 16	3
201_	2	0	1	10, 10, 20	3
120_	1	2	0	10, 16, 16	3
021_	0	2	1	16, 16, 20	3
102_	1	0	2	10, 20, 20	3
012_	0	1	2	16, 20, 20	3
111_	1	1	1	10, 16, 20	3

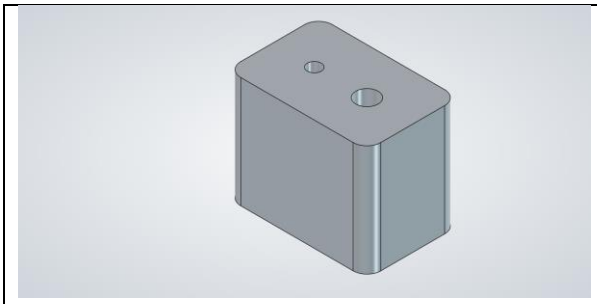
Note that for this example, pens are placed first, then ballpoints, then markers. A different order could be used, but it must be stable and documented.

The Configuration Key is valid only if it exists in the table. The model should display a clear diagnostic status rather than producing partial or misleading geometry.



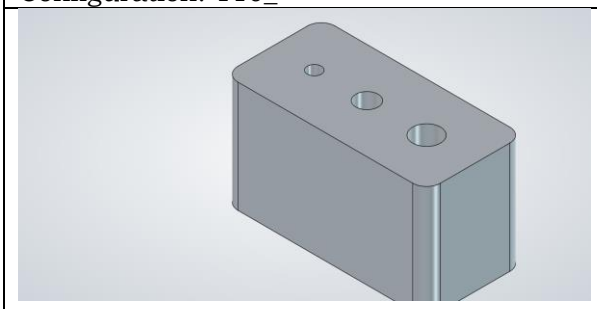
	A	B	C	D
1	**Parameter**	**Alias**	**Value**	
2	# of pens	n_pen	1	[Choose between 0, 1, 2, 3]
3	# of ballpoints	n_ballpoint	0	[Choose between 0, 1, 2, 3]
4	# of markers	n_marker	0	[Choose between 0, 1, 2, 3]
5		n_checksum	1	INVALID if not 1, 2 or 3
6		n_tot	1	OK
7				
8	Thumb Width	thumb_width	25,00 mm	Determines spacing between p

Configuration: 100_



	A	B	C	D
1	**Parameter**	**Alias**	**Value**	
2	# of pens	n_pen	1	[Choose between 0, 1, 2, 3]
3	# of ballpoints	n_ballpoint	1	[Choose between 0, 1, 2, 3]
4	# of markers	n_marker	0	[Choose between 0, 1, 2, 3]
5		n_checksum	2	INVALID if not 1, 2 or 3
6		n_tot	2	OK
7				
8	Thumb Width	thumb_width	25,00 mm	Determines spacing between positions

Configuration: 110_



	A	B	C	D
1	**Parameter**	**Alias**	**Value**	
2	# of pens	n_pen	1	[Choose between 0, 1, 2, 3]
3	# of ballpoints	n_ballpoint	1	[Choose between 0, 1, 2, 3]
4	# of markers	n_marker	1	[Choose between 0, 1, 2, 3]
5		n_checksum	3	INVALID if not 1, 2 or 3
6		n_tot	3	OK
7				
8	Thumb Width	thumb_width	25,00 mm	Determines spacing between positions

Configuration: 111_

Figure C–2: Three configurations driven by the user input. The light blue cells are the parameters the user selects in a separate Spreadsheet (UIParams). The Spreadsheet that holds the engineering logic (CMParams, see 0) generates the Configuration Key, and the model follows.

Simplified configuration-table demonstrator. The geometry is not a fully engineered or validated product; it omits fillets, chamfers, tapered or relieved pocket holes, material-balance considerations, stress-relief features and manufacturing allowances.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	config_key	n_pen	n_ballpoint	n_marker	hole_sequence	n_tot_eff	d_1	d_2	d_3	x_1	x_2	x_3	length	width
2	201_	2	0	1	PPM	3	10,00 mm	10,00 mm	20,00 mm	30,00 mm	65,00 mm	105,00 mm	140,00 mm	70,00 mm
3	001_	0	0	1	<<M>>	1	20,00 mm	0,00 mm	0,00 mm	35,00 mm	0,00 mm	0,00 mm	70,00 mm	70,00 mm
4	002_	0	0	2	<<MM>>	2	20,00 mm	20,00 mm	0,00 mm	35,00 mm	80,00 mm	0,00 mm	115,00 mm	70,00 mm
5	003_	0	0	3	<<MMM>>	3	20,00 mm	20,00 mm	20,00 mm	35,00 mm	80,00 mm	125,00 mm	160,00 mm	70,00 mm
6	010_	0	1	0	<<B>>	1	16,00 mm	0,00 mm	0,00 mm	33,00 mm	0,00 mm	0,00 mm	66,00 mm	66,00 mm
7	011_	0	1	1	<<BM>>	2	16,00 mm	20,00 mm	0,00 mm	33,00 mm	76,00 mm	0,00 mm	111,00 mm	70,00 mm
8	012_	0	1	2	<<BMM>>	3	16,00 mm	20,00 mm	20,00 mm	33,00 mm	76,00 mm	121,00 mm	156,00 mm	70,00 mm
9	020_	0	2	0	<<BB>>	2	16,00 mm	16,00 mm	0,00 mm	33,00 mm	74,00 mm	0,00 mm	107,00 mm	66,00 mm
10	021_	0	2	1	<<BBM>>	3	16,00 mm	16,00 mm	20,00 mm	33,00 mm	74,00 mm	117,00 mm	152,00 mm	70,00 mm
11	030_	0	3	0	<<BBB>>	3	16,00 mm	16,00 mm	16,00 mm	33,00 mm	74,00 mm	115,00 mm	148,00 mm	66,00 mm
12	100_	1	0	0	<<P>>	1	10,00 mm	0,00 mm	0,00 mm	30,00 mm	0,00 mm	0,00 mm	60,00 mm	60,00 mm
13	101_	1	0	1	<<PM>>	2	10,00 mm	20,00 mm	0,00 mm	30,00 mm	70,00 mm	0,00 mm	105,00 mm	70,00 mm

Figure C–3: The Configuration Table used for the pen holder (partly shown). Row #2 (shown in yellow) contains the derived values and calculations generated by the inputs from the user (in this selection: 2 pen holes and 1 marker hole generated config_key 201_).

© 2026 Magimo

Appendix D Bit holder: design basis and calculations

D.1 Purpose

The running example in Chapters 2–7 shows how the bit holder is set up as a parametric model. This appendix shows what lies behind it: the design basis, the calculations, the assumptions and the issues that are still open. It is arranged as an extract of a Technical File (§7.1.1), so it doubles as an example of what such a file records for a real part.

The extract has status Draft (§6.1). Several values are starting assumptions that the tests in §6.3 must confirm or replace. That is deliberate: a Technical File records what is known and what is not, rather than only the finished answer.

D.2 Design intent and requirements

The holder stores 1/4″ hex screw bits on a French cleat in a workshop or garage. A deeper row next to the wall takes quick-change holders or double-ended bits. To change bits, the user lifts the holder off the cleat and works at the bench. The full brief is given in the running-example box in §2.1. The requirements that drive the calculations are:

- the holder lifts off and goes back on without tools;
- bits stay in when the holder is carried or tilted;
- the hook carries the loaded holder over time without creeping off the cleat;
- the holder fits the usable build envelope of the user's printer, printed with its top face on the bed.

D.3 Assumptions and design boundaries

The calculations use configuration 4 × 10 (`rows_across × rows_out`) with the holder row, 25 mm bits, PETG and normal fit: the longest configuration in the boundary tests (§6.2).

Table D–1: Assumptions and design boundaries (4 × 10 configuration)

Quantity	Value	Basis
Bit shank	6.35 mm across flats; 7.33 mm across corners	1/4″ hex standard
Bit mass, 25 mm	6 g	Steel; weigh to confirm
Socket, normal fit	6.60 mm AF; 7.62 mm AC	+0.25 mm clearance, e.g.; coupon §6.3.6
Socket depth; floor	8 mm; 2 mm	Clamp 8–12 mm; 10 layers
Holder-row depth	20 mm	Quick-change holders and double-ended bits
`min_wall`	1.8 mm	4 × 0.45 mm line width
`bit_spacing`	6.5 mm, edge to edge	Finger access, e.g.; grip test to confirm
Edge; heel; holder-row gap	6 mm; 7.5 mm; 5 mm	e.g.
Plate angle	15°	Presentation angle; see D.4.1
Front thickness	10 mm minimum	Clamp; see D.4.1

© 2026 Magimo

Quantity	Value	Basis
Height	40–52 mm	`hook_depth` (40 mm) sets the minimum
Hook	Tip against the wall; root ligament 12.1 mm; R 2 mm dogbone	Sketch
Wall cleat	18 mm plywood, 45°; rear face 0.5 mm clear of the cleat	Fixed context
Material	PETG, 1.27 g/cm³; fill factor 0.5	Data sheet; 4 perimeters, 20 % infill
Quick-change holder	35 g each, a full row	e.g.; weigh to confirm
Push on a bit (misuse)	30 N at the outer row	e.g.; to be measured
Friction at the contacts	Not calculated	See Hook on the cleat
Build volume	Entered by the user; `bed_margin` 5 mm per side	Usable envelope = nominal − 2 × `bed_margin`

D.4 Design rationale and calculations

D.4.1 Layout

- A 15° inclination is chosen as a practical presentation angle for small parts that are picked often. It improves visibility and access; comparable industrial bin rails and picking shelves use a 15° tilt for the same purpose.
- Because the underside is inclined, the height follows from the number of rows out: `y_top = t_front + run · tan(15°)`, with `t_front = max(socket_depth + floor_t; hook_depth − run · tan(15°))`. Up to 6 rows the hook sets the height at 40 mm; beyond that the slope does (52 mm at 10 rows).
- `socket_ac` = (6.35 + 0.25) / cos(30°) = 7.62 mm.
- `pitch = socket_ac + max(bit_spacing; min_wall) = 7.62 + max(6.5; 1.8)` = 14.12 mm.
  Finger access, not the socket wall, sets the pitch. The rule is applied to orthogonal neighbours, which is conservative; see the footnote 4 to the §4.5 running example box for the diagonal alternative.
- `plate_width` = (4 − 1) × 14.12 + 7.62 + 2 × 6 = 62.0 mm. For a printer with a 220 mm nominal bed width, the usable width is 210 mm. The largest count across is then 14: 13 × 14.12 + 7.62 + 12 = 203 mm, where 15 would need 217 mm.
- `x_bits1` = 18.5 + 6 + 7.62/2 + 14.12 + 5 = 47.4 mm (rear face, edge, holder row, pitch, gap); `x_front` = 47.4 + (10 − 1) × 14.12 + 7.62/2 + 6 = 184 mm.

D.4.2 Loads and moments

- `m_holder` ≈ 225 g (fill factor 0.5; solid PETG would be 450 g)
- `m_bits` = 40 × 6 g = 240 g; `m_qh` = 4 × 35 g = 140 g
- W = (225 + 240 + 140) g × 9.81 m/s² = 5.9 N, acting x = 88 mm from the wall
- `moment_self` = W · x = 0.52 Nm (the same formula as `MomentPass` in CMParams)
- `x_outer` = 47.4 + 9 × 14.12 = 175 mm; a 30 N push there adds 30 N × 0.175 m = 5.2 Nm

Pushing a bit into the outer row produces about ten times the moment of the filled holder's own weight. The holder row next to the wall adds 0.04 Nm (7 % of `moment_self`); the same four holders in the outer row would add 0.24 Nm (46 %). Putting the heaviest items next to the wall matters.

D.4.3 Hook on the cleat

The hook tip touches the wall; the rear face stays 0.5 mm clear of the cleat. The weight tips the holder forwards about the cleat ridge, which sits in the dogbone. The tip then presses against the wall. Two forces from the wall system hold the holder: on the bevel at the ridge, a force N perpendicular to the 45° bevel; at the hook tip, a horizontal force P from the wall.

Moment about the tip contact, with $\Delta x + \Delta y = 31$ mm the distance terms of the ridge relative to the tip (fixed by the hook geometry):

- $P = W \cdot x / (\Delta x + \Delta y)$
- $N = \sqrt{2} \cdot P$

Friction is not calculated. It acts at both contacts and has two effects. First, it holds the hook down: the vertical part of N is larger than W (2.8 × W at 4 × 10), and friction takes the difference, so without friction the hook would ride up the bevel. The retention test on the cleat confirms that the holder stays on (D.5). Second, friction at the tip adds some bending at the hook root, up to about 50 %. The stresses below include that margin.

The critical point is the dogbone. Section A–A runs from the dogbone hole to the hook top, perpendicular to the bevel: ligament $t = 20\ \text{mm}/\sqrt{2} - 2\ \text{mm} = 12.1$ mm, width $b = 62$ mm, section modulus $Z = b \cdot t^2/6 = 1520\ \text{mm}^3$. The main load on it is P at the tip, about 21 mm below the centre of A–A. Stress at the dogbone: $\sigma = Kt \cdot M/Z$, with M the moment of P and N about the centre of A–A, and $Kt \approx 2$ for the R 2 mm hole (estimate).

This is a simplified model: rigid holder, contact at the ridge and at the lowest point of the tip face. It gives orders of magnitude, not design values.

D.4.4 What the numbers mean

Table D–2: Contact forces and dogbone stress for three load cases

Case	W	x	N	P	σ at the dogbone
4 × 4, own weight	3.4 N	49 mm	8 N	5 N	0.2 MPa
4 × 10, own weight	5.9 N	88 mm	24 N	17 N	0.6 MPa
4 × 10, 30 N push at the outer row	36 N	160 mm	260 N	185 N	7 MPa

- Strength is not critical. Printed PETG reaches roughly 45–50 MPa along the layers and about half across them. Own weight uses less than 3 % of the weaker direction; the short push case about 30 %. The hook bends across the layers in the released orientation (§5.4), which these margins allow.
- Storage is the normal case, and creep governs it. The holder hangs for months under its own weight. The sustained moment slowly creeps the hook root and the contacts, so the holder tilts further forwards over time. The creep test (§6.3) measures that tilt and sets `allowable_moment` from evidence rather than from this estimate.
- Pushing bits in while the holder hangs is the case to avoid. The contact forces rise about ten times, and the hook relies more on friction to stay on the cleat. That is the reason to lift the holder off before inserting bits. Lifting off itself is not critical: the user grips the holder close to the heel and lifts almost vertically, which cancels the moment.

© 2026 Magimo

- The forces grow with the lever x. More rows out mean a longer lever, so the limit on `rows_out` protects the creep margin, not the strength.
- The dogbone keeps the cleat ridge off a sharp corner, where the contact pressure is highest. Its Kt is an estimate; the creep test checks the root for white stress marks.

D.5 Validation evidence

Table D–3: Validation tests and their status

Test	Purpose	Status
Hex tolerance-ladder coupon (§6.3.6)	Socket clearance for PETG, sockets vertical	Planned
Retention and insertion (§6.3)	Bits stay in when carried; push force	Planned
Retention on the cleat	The holder stays on the cleat, filled, and with a bit pushed in at the outer row	Planned
Creep test (§6.3)	`allowable_moment`; forward tilt and hook root at 40 °C over 7 days	Planned
Grip test	`bit_spacing`; orthogonal or diagonal access	Planned
Boundary and coupled-extreme tests (§6.2)	Model behaviour across the supported domain	Planned

Until these tests are recorded, the default configuration is the only candidate for Supported status, and holders with more than e.g. 6 rows out remain generated but not Supported.

D.6 Limitations and open issues

Table D–4: Open issues and how they are resolved

Issue	Effect	Resolved by
Socket clearances are starting values	Retention not proven	Coupon (§6.3.6)
`allowable_moment` estimated, not measured	`rows_out` above e.g. 6 not Supported	Creep test
Friction not calculated	Margin against riding up under a push not proven	Retention on the cleat
Only 18 mm cleats supported	Other wall systems not covered	Advanced option once tested (§3.6.2)
Double-ended bits in the holder row	Fit of reduced or grooved hex bodies unknown	Trial in the first print
Free space needed above the holder	Lift-off may be blocked by items above	Stated in the use instructions; value to confirm

© 2026 Magimo

D.7 Revision history

Revision	Date	Change
0.1	2026-10-30	First draft for the running example; calculations based on assumptions in D.3.
0.2	2026-11-11	Updated to model v5d: PETG only, 18 mm cleat, 25 mm bits, holder row, orientation decided. Hook calculation without friction.

© 2026 Magimo

Appendix E Advanced FreeCAD Toolbox

E.1 Purpose

This appendix is a practical toolbox for advanced FreeCAD users who build parametric models intended to survive change, manufacturing, handover and repeated configuration. The main chapters establish the design principles: expose meaningful decisions, centralise relationships, protect engineering logic, encode manufacturing constraints, validate the supported domain and release only controlled configurations. Appendix B and Appendix C describe the main control-layer and configuration-table structures respectively. This appendix focuses on the smaller techniques that make those principles work in practice.

Many of the techniques are easy to overlook. They include expression functions, Spreadsheet patterns, property links, lookup methods, diagnostic cells, geometry strategies and workflow shortcuts that can make a substantial difference to model robustness. Examples include:

- using `floor()`, `ceil()` and `round()` to convert continuous dimensions into whole numbers of layers, beads, ribs, holes, passes or modules;
- using `min()`, `max()` and `abs()` to protect relationships without hiding invalid input;
- keeping requested, derived, actual and status values separate;
- using aliases and controlled identifiers instead of fragile raw cell references;
- deriving feature counts and spacing from available boundaries;
- driving clearances, wall thicknesses and feature sizes from process data;
- using bounding-box and shape properties for automatic checks;
- separating optional geometry from the stable model core;
- coordinating multiple Bodies through a controlled parameter structure; and
- making validation, active-result selection, export and release status explicit.

The patterns are generic. They can be applied to additive manufacturing, sheet metal, machining, consumer products, mechanisms, jigs and fixtures, enclosures and other parametric designs. The examples are deliberately DfAM-oriented because additive manufacturing makes poor parametrisation visible through failed prints, fused interfaces, unsupported features, distortion, weak details and dimensional errors. The same reasoning often applies to other manufacturing processes.

This appendix is not a collection of universal FreeCAD recipes. Expressions, property names, supported functions and Python interfaces may vary with the FreeCAD version, workbench, object type and expression context. Test each pattern in the target environment before incorporating it into a released template. Use the appendix as a menu of tools:

- select the pattern that addresses the current design problem;
- adapt names, aliases, units and sheet structures to the model;
- keep expressions short enough to inspect and debug;
- separate implementation values from user-facing messages;
- add explicit checks where a shortcut could conceal an invalid result; and
- record the assumptions and test evidence required before release.

© 2026 Magimo

A useful pattern should do more than make the model generate geometry. It should make the model's behaviour easier to understand, test, manufacture, configure or hand over. The objective is not to collect clever tricks for their own sake. It is to use small, well-understood techniques to build a parametric model whose permitted behaviour remains visible and justifiable after it leaves its creator.

E.2 Spreadsheet architecture that scales

E.2.1 When to split the control layer

Appendix B defines the individual sheets in the recommended Spreadsheet control layer: UIParams, CMParams, MaterialProfile, ProcessAspects, UseConstraints, Validation, ReleaseInfo and DocsIndex. This section addresses a different question: when that structure should be divided into separate sheets and when a simpler arrangement is sufficient.

The right architecture is the smallest one that keeps responsibility, dependency and status unambiguous. Do not add a sheet merely because a value could be stored there. Add one when separation makes the model easier to understand, review, test or maintain.

E.2.1.1 Start simple

For a small internal model, a single Spreadsheet with clearly separated sections is often sufficient. Use clearly labelled areas for:

- user inputs and choices;
- protected engineering values;
- calculated geometry-driving values;
- validation and status information.

This arrangement is appropriate when the model has few user-facing decisions, one material or process context, limited validation evidence and no external automation. It keeps the control layer compact and easy to inspect.

A single sheet becomes unsuitable when its sections begin to obscure the distinction between user inputs, engineering rules and calculated results. If a reviewer must scroll through mixed content to determine which value is authoritative, split the sheet.

E.2.1.2 Split by responsibility

Separate a sheet when one of the following conditions applies:

- The model has distinct user-facing choices and substantial creator-controlled logic.
- Material data, process assumptions or use constraints are maintained independently of the geometry.
- Several people maintain different parts of the parameter system.
- The model supports multiple materials, printers, process profiles or configuration families.
- Validation evidence must be associated with specific configurations, profiles or model revisions.
- The model is released, reused, automated or handed to another maintainer.
- A sheet has become difficult to navigate, audit or protect against accidental edits.

© 2026 Magimo

The purpose of separation is not to create a larger model. It is to make each sheet answer one clear question: what does the user provide, what does the creator control, what does the model calculate and what evidence supports the result?

E.2.1.3 Keep the dependency direction one-way

Whether the control layer uses one sheet or several, information should flow in one direction. Geometry should normally consume creator-controlled or derived aliases, not raw user cells. This keeps the engineering interpretation of a user input in one place and prevents the same requirement from being implemented differently in several sketches.

Avoid reverse dependencies. Do not read a geometry-derived result back into the sheet or sketch that drives the geometry that produced it. If a geometry-derived value is needed for reporting, place it in a separate diagnostics or results layer.

A typical dependency direction is:

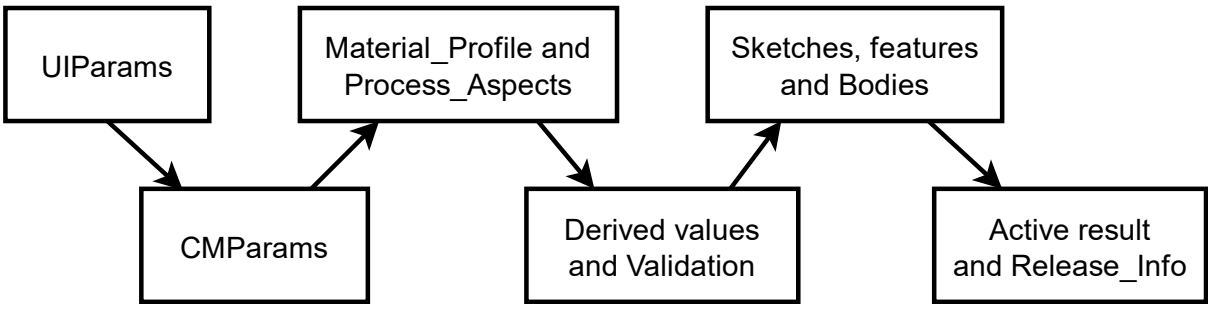


Figure E–1: Practical dependency rule

E.2.1.4 Combine sheets only when clarity survives

Combining sheets is acceptable when the resulting layout remains immediately understandable. For example, a small model may combine material and process data in one profile section or combine validation and release information in one status area. Do not combine sheets when doing so makes it harder to identify:

- the authoritative source of a value;
- the boundary between user and creator responsibility;
- the dependency direction;
- the validation status of a configuration;
- the version and compatibility of a released result.

When in doubt, separate the sheets. A clear structure is easier to extend than a compact structure that has become ambiguous.

E.2.1.5 Practical rules

- For the purpose and contents of each sheet: see Appendix B.
- Start with the simplest structure that keeps responsibility and dependency clear.
- Use one sheet with separated sections for small, self-contained models.
- Split sheets when user configuration, engineering logic, profiles, validation or release control become substantial.
- Keep the dependency direction one-way: inputs to engineering logic to geometry to validation.

© 2026 Magimo

- Let geometry consume controlled or derived aliases, not raw user inputs.
- Do not create a sheet unless it improves clarity, review, testing or maintainability.
- Review the architecture whenever the model gains new user choices, profiles, validation requirements or external interfaces.

For anything beyond a simple part, divide the control layer according to responsibility. The following template is a practical starting point; omit sheets that do not serve a real purpose and add sheets only when they make the model easier to understand or control.

E.2.2 Process parameters for FDM geometry rules

E.2.2.1 Problem

Geometry rules for walls, ribs, corrugations, teardrops, clearances and living hinges all depend on a small set of process parameters. If these are hard-coded per part, it becomes difficult to maintain and validate the model.

E.2.2.2 Trick

Define a small set of shared process parameters that all geometry rules can reference. Typical entries in ProcessAspects or CMParams include:

Table E–1: Typical process parameters for FDM printing

Parameter	Meaning	Example value
`w_extrusion`	Effective extrusion width (bead width)	0.45 mm for 0.4 mm nozzle
`h_layer`	Layer height	0.15–0.20 mm
`t_min_tech`	Minimum wall thickness for the selected technology	1.2 mm for FDM with a 0.4 mm nozzle
`n_min_wall`	Minimum number of extrusion lines for functional walls	2–3
`alpha_selfsupport`[7]	Maximum self-supporting overhang angle	45°, process-dependent

These values should be tied to the active material and process profile, not hard-coded per part.

E.2.2.3 DfAM example

A parametric enclosure uses `w_extrusion`, `h_layer`, `t_min_tech` and `alpha_selfsupport` from the active process profile to derive wall thickness, rib dimensions and corrugation pitch.

Other uses

- Teardrop profiles
- Orientation-dependent clearances
- Living hinges and flexures
- Hardware pockets

7 Greek characters may be used in display names and mathematical descriptions, but are invalid in aliases in FreeCAD 1.1.2. Use an ASCII alias such as `alpha_selfsupport` for reliable use in expressions, exports and external interfaces.

© 2026 Magimo

E.2.2.4 Watch-outs

- Do not treat these values as universal constants; they depend on material, machine and process profile.
- Keep them in a controlled, versioned profile sheet.

E.3 Expression patterns that matter in any domain

FreeCAD expressions are most valuable when they make a design rule explicit, reusable and testable. Their purpose is not to produce the cleverest possible one-line formula. A good expression converts a meaningful input into a controlled result that can be inspected by a future maintainer.

> **Do not flatten engineering logic into opaque one-cell formulas**
>
> Combining many calculations into one long expression can reduce the number of cells, but it also hides the engineering reasoning. Prefer a small number of named intermediate values with meaningful aliases. A readable chain such as fitClearance → outerDiameter → wallCheck is easier to review, test and maintain than a single nested formula. Keep chains short, but do not eliminate intermediate values merely to make the dependency graph look smaller.

Use the Spreadsheet control layer described in Chapter 3 and Appendix B to hold the engineering logic. Sketches and features should normally consume named, derived values rather than contain long calculations or raw cell references. The guide's central expression rule still applies: one important relationship should have one authoritative definition. This section collects small expression patterns that are especially useful in parametric models:

- Guard an output against an impossible or unsafe value.
- Snap an input to a manufacturing increment or standard size.
- Count repeated features as whole numbers rather than fractional geometry.
- Reserve enough remaining space for end margins, walls, radii or tool access.
- Compare a calculated result with a required limit.
- Expose the difference between requested and actual values.
- Report a clear status without mixing it into geometry-driving formulas.

A practical implementation uses several short cells instead of one opaque expression. For example, do not place a complex calculation directly in the length constraint of a sketch. Define the relevant inputs and intermediate results in CMParams, then reference the final alias:

- `requested_length`
- `end_margin`
- `available_length`
- `feature_pitch`
- `feature_count`
- `actual_pitch`
- `feature_status`

The geometry can then use `feature_count` and `actual_pitch`, while the user-facing or release-facing logic reads `feature_status`.

Expression conventions

Use these conventions:

- Use aliases rather than raw addresses such as `Spreadsheet.B12`.
- Keep physical quantities unit-bearing: 2.4 mm, not a bare 2.4 whose meaning may have to be guessed.
- Keep counts unitless and separate from the dimensions they control.
- Name intermediate values after their engineering role, not their calculation order.
- Make the distinction between requested, derived, actual and status values visible.
- Prefer a diagnostic that identifies the failed relationship over a silent clamp that merely generates geometry.
- Test functions and syntax in the target FreeCAD release; availability and behaviour may differ by version, object type or expression context.

The following subsections develop the most reusable patterns: guardrails with `min()`, `max()` and `abs()`; snapping with `floor()`, `ceil()` and `round()`; tolerance-stack checks and controlled directional compensation.

E.3.1 Guardrail functions: `min()`, `max()` and `abs()`

E.3.1.1 Problem

A derived dimension can become too small, too large or negative when related parameters change. That may lead to failed features, inverted geometry, impossible clearances or results that remain mathematically valid but violate the design rules.

E.3.1.2 Trick

Use `max()` to enforce a lower bound, `min()` to enforce an upper bound and `abs()` where only the magnitude of a signed difference matters. FreeCAD supports `min()`, `max()` and `abs()` in expressions.

```
EffectiveWall = max(RequestedWall; MinimumWall)

PocketDepth = min(RequestedDepth; AvailableDepth)

OffsetMagnitude = abs(MeasuredOffset)
```

To keep a value within both limits:

```
ControlledValue = min(MaxAllowed; max(MinAllowed; RequestedValue))
```

E.3.1.3 DfAM example

A wall needs at least three extrusion lines:

```
MinimumWall = 3 * LineWidth

EffectiveWall = max(RequestedWall; MinimumWall)

WallDelta = EffectiveWall - RequestedWall
```

If `LineWidth` is 0.45 mm and the requested wall is 1.20 mm, the effective wall becomes 1.35 mm.

Keep the adjustment visible:

```
WallStatus = WallDelta > 0mm ? <<WARNING: wall increased>> : <<OK>>
```

E.3.1.4 Other uses

- Keep a sheet-metal bend radius above a qualified minimum
- Prevent a pocket from cutting through its remaining base
- Limit a machining depth to accessible tool length
- Maintain a minimum edge distance around a hole
- Report either direction of a measurement error as a positive deviation

E.3.1.5 Watch-outs

- A clamp changes the result; it does not prove that the geometry is valid.
- Do not silently adjust a user-defined functional interface. Store the requested and actual values separately.
- Do not use `abs()` if direction matters, such as left/right placement or inside/outside offset.
- Compare like with like: lengths with lengths, angles with angles and unitless counts with counts.
- Use a separate status check when the configuration should fail rather than be automatically corrected.

E.3.2 Snapping to process steps: `floor()`, `ceil()` and `round()`

E.3.2.1 Problem

User inputs are usually continuous, but manufacturing often works in discrete increments: layers, extrusion lines, tool passes, modular lengths, standard stock sizes, ribs, holes or pattern repetitions.

E.3.2.2 Trick

As a general principle, divide the desired dimension by the process increment, convert the result into an integer, then multiply it back by the increment. FreeCAD supports `ceil()`, `floor()` and `round()` for this purpose. See the FreeCAD Expression documentation.

```
StepCount = ceil(RequestedDimension / ProcessIncrement)

ActualValue = StepCount * ProcessIncrement
```

Table E–2: Snapping to process steps (examples)

Requirement	Count expression	Result
Must meet or exceed the request	`=ceil(Requested / Increment)`	Actual value is equal to or above the request
Must not exceed available space	`=floor(Requested / Increment)`	Actual value is equal to or below the request
Nearest practical increment	`=round(Requested / Increment)`	Actual value may move in either direction

Corrugation pitch snapping

For repeated surface features such as corrugations, it is often useful to snap the pitch to integer multiples of layer height. A practical implementation is:

1. Define a requested pitch `p_req`.
2. Compute an effective pitch, `p_eff`.
3. Use `p_eff` to drive the geometry.

E.3.2.3 DfAM example

Make a feature height an exact multiple of layer height:

```
RequestedHeight = 17.25 mm

LayerHeight = 0.20 mm

LayerCount = ceil(RequestedHeight / LayerHeight)

ActualHeight = LayerCount * LayerHeight

HeightDelta = ActualHeight - RequestedHeight
```

The model derives 87 layers and an actual height of 17.40 mm. The result is deliberately above the requested height because this particular feature must not fall short.

E.3.2.4 Other uses

- Choose an integer number of machining passes from the permitted step-down
- Fit a whole number of ribs, teeth, holes or ventilation slots
- Cut standard stock lengths within an available space
- Create modular rails, panels, drawers or enclosure sections
- Round feature depths to approved manufacturing increments

E.3.2.5 Watch-outs

- Calculate a unitless count first, then return to a dimensional result.
- `ceil()` can exceed a build envelope; `floor()` can produce an undersized feature.
- Keep the requested value, count, actual value and deviation in separate cells.
- The increment must be positive and non-zero.
- Do not use `round()` for a safety, sealing, fit or load-critical requirement unless either rounding direction is acceptable.

E.3.3 Snap wall thickness to process multiples

E.3.3.1 Problem

An arbitrary CAD wall thickness can lead to partial perimeter lines, inconsistent infill contact, thin-wall artefacts or unpredictable strength. The issue often appears when wall thickness is treated as an unrestricted user dimension rather than a process-derived result.

E.3.3.2 Trick

Calculate the number of process lines first, then derive the actual wall thickness:

```
PerimeterCount = max(MinPerimeterCount; ceil(RequestedWall /
LineWidth))

ActualWall = PerimeterCount * LineWidth
```

If the wall must fit inside a fixed geometry range, calculate whether enough space exists:

```
MaxPerimeterCount = floor(AvailableWallSpace / LineWidth)

WallStatus = MaxPerimeterCount >= MinPerimeterCount ? <<OK>> :
<<FAIL>>
```

E.3.3.3 DfAM example

For a 0.45 mm effective line width and a requested wall of 1.10 mm:

```
PerimeterCount = ceil(1.10mm / 0.45mm)

ActualWall = PerimeterCount * 0.45mm
```

The result is three perimeter lines and an actual wall of 1.35 mm.

E.3.3.4 Other uses

- Select a sheet thickness from available gauges
- Build a laminate from an integer number of plies
- Set a web thickness from approved stock or tool-access increments
- Define moulded wall families from qualified thicknesses
- Choose an integer number of shims, spacers or gasket layers

E.3.3.5 Watch-outs

- An integer number of lines does not guarantee print quality; slicer behaviour may include variable line width or thin-wall modes.
- Use the effective line width from the active process profile, not simply the nominal nozzle diameter.
- Do not treat perimeter count as a structural calculation.
- Check that a thicker wall does not collide with nearby cavities, fasteners or mating features.
- If the external envelope is fixed, reject the configuration or redesign the surrounding geometry instead of silently increasing the wall.

E.3.4 Checking alignment with mod()

Snapping a value to a process multiple produces a value that should align with the intended process step. The `mod()` function can check how far a value is from a whole number of those steps. It returns the remainder after dividing x by y. For example, `mod(7; 2)` returns 1. The function is invalid when the divisor is zero.

To check whether a wall thickness aligns with a whole number of extrusion lines:

```
=mod(WallThickness; LineWidth)
```

Assign this the alias `WallRemainder`.

- For a wall thickness of 2.4 mm and a line width of 0.4 mm, the remainder is 0. The wall aligns with six extrusion lines.
- For a wall thickness of 2.5 mm and a line width of 0.4 mm, the remainder is 0.1 mm. The wall does not align with a whole number of lines.

Because calculated values can be very close to, but not exactly equal to, zero, compare the remainder with a tolerance rather than expecting an exact zero:

```
=or(WallRemainder <= AlignmentTolerance; WallRemainder >= LineWidth -
AlignmentTolerance)
```

Assign this the alias `WallAlignmentPass`.

Alias	Example value
WallThickness	2.5 mm
LineWidth	0.4 mm
AlignmentTolerance	0.02 mm

The remainder is 0.1 mm. This is greater than the tolerance and smaller than `LineWidth - AlignmentTolerance`, so `WallAlignmentPass` is 0. A status cell can report the result:

```
=WallAlignmentPass ? <<WALL ALIGNED WITH LINE WIDTH>> : <<WALL NOT
ALIGNED WITH LINE WIDTH>>
```

Practical rules

- Use `mod()` to detect whether a value aligns with a process multiple.
- Guard against a zero divisor before using `mod()`.
- Compare the remainder with a tolerance; do not expect an exact zero from calculated values.
- Keep the remainder, tolerance and pass/fail result as separate named values.
- Use the check to identify misalignment, not to silently replace a user value.
- Review the alignment rule when the nozzle, extrusion width, layer height or material profile changes.
- Snapping produces the intended value; a separate alignment check confirms that the result remains consistent with the process step.

E.3.5 Tolerance-stack checks

A tolerance stack combines the variation of several dimensions, interfaces or process effects into one overall variation. The relevant contributors, calculation method, permitted limit and pass/fail result should remain visible in the model. Use the simplest structure that makes the engineering decision clear. A small number of stable contributors may be combined directly. A larger or evolving stack is easier to maintain as a range of named contributors.

E.3.5.1 Linear stack from explicit contributors

When the number of contributors is small and stable, the clearest implementation is a direct linear calculation using named values. Each contributor should have a meaningful alias and a defined engineering role. The total stack is then calculated from those contributors and compared with a declared permitted limit. Table E–3 gives a typical arrangement.

Table E–3: Tolerance-stack checks

Alias	Example value	Meaning
BaseGrowth	0.12 mm	Dimensional variation of the base part
LidGrowth	0.10 mm	Dimensional variation of the lid part
AlignmentVariation	0.06 mm	Variation caused by location or assembly
LinearStack	`=BaseGrowth + LidGrowth + AlignmentVariation`	Worst-case combined variation
PermittedStack	0.40 mm	Declared functional limit
LinearStackPass	`=LinearStack <= PermittedStack`	Pass/fail result

For the example values, the linear stack is 0.28 mm. Because this is below the permitted stack of 0.40 mm, `LinearStackPass` is 1.

A linear stack assumes that the contributors may all act in the same unfavourable direction. It is therefore the conservative result. Use this method for fits, seals, locating interfaces, press fits and other situations where the combined variation must remain controlled in every produced part.

The calculated stack and the permitted stack should remain separate values. The calculated stack describes the current design; the permitted stack states the functional requirement. Combining them into one formula makes the design decision harder to inspect and change.

A status cell can report the result:

```
=LinearStackPass ? <<LINEAR STACK PASS>> : <<REVIEW TOLERANCE STACK>>
```

This status describes whether the declared stack rule is satisfied. It does not by itself establish that the configuration has been printed, measured or functionally validated.

Practical rules

- Give every contributor a meaningful alias.
- Keep each contributor in its own named cell.
- Use consistent units throughout the stack.
- Calculate the stack from named contributors, not from raw cell addresses.
- Keep the calculated stack and permitted stack separate.
- Use a linear stack when contributors may act unfavourably together.
- Test the stack at nominal, minimum and maximum contributor values.
- Review the stack whenever an interface, material, process or fit rule changes.

© 2026 Magimo

E.3.5.2 Range-based tolerance stacks

What it does

A tolerance stack combines the dimensional variation of several features, interfaces or process effects into one overall variation. In a parametric model, the stack should not be hidden inside a single long expression. It should be built from named contributors that can be inspected, revised and tested.

FreeCAD aggregate functions accept cell ranges. This makes it possible to treat a tolerance stack as a column of contributors:

- each contributor is one named value;
- each contributor has a clear engineering meaning;
- adding a contributor means adding a row;
- the total stack is calculated from the visible contributor range.

This keeps the calculation readable and makes it easier to see which interface, feature or process assumption dominates the result.

Why it matters

A tolerance stack affects whether parts assemble, slide, seal, locate or remain within the intended functional conditions of use. It is rarely enough to check one dimension in isolation. For example, a lid may fit even when the base is slightly oversized, but fail when base growth, lid shrinkage, seam clearance and fastener-position variation all act in the same direction. A range-based stack makes those contributors visible before they are combined into geometry.

A stack is also a design-review tool. If one contributor is much larger than the others, it identifies where attention is needed: reduce the variation, change the interface, add a compensation rule or redefine the fit.

Basic structure

Use a dedicated block in the creator-control or validation layer.

Table E–4: Range-based stack block (example)

Alias	Formula / value	Meaning
Contribution1	0.10 mm	Base dimensional variation
Contribution2	0.15 mm	Lid dimensional variation
Contribution3	0.08 mm	Seam or interface variation
LinearStack	`=sum(Contribution1:Contribution3)`	Worst-case linear stack
Square1	`=Contribution1^2`	Squared contributor
Square2	`=Contribution2^2`	Squared contributor
Square3	`=Contribution3^2`	Squared contributor
StatisticalStack	`=sqrt(sum(Square1:Square3))`	Root-sum-square estimate
PermittedStack	0.40 mm	Declared functional limit
LinearStackPass	`=LinearStack <= PermittedStack`	Worst-case check

Alias	Formula / value	Meaning
StatisticalStackPass	`=StatisticalStack <= PermittedStack`	Statistical check

For the example values:

- `LinearStack` = 0.33 mm
- `StatisticalStack` = approximately 0.20 mm
- Both checks pass against a permitted stack of 0.40 mm.

The values are illustrative. They do not represent universal material, process or fit limits.

Linear and statistical stacks

Use the linear stack when every contributor can move in the same unfavourable direction. It is the conservative result:

```
=Contribution1 + Contribution2 + Contribution3
```

Use the statistical stack only when the contributors are reasonably independent and the assumed distribution is appropriate for the decision being made. The common root-sum-square form is:

```
=sqrt(Contribution1^2 + Contribution2^2 + Contribution3^2)
```

In the Spreadsheet, calculate each square in its own cell and sum the range:

```
=sqrt(sum(Square1:Square3))
```

The statistical result is smaller than the linear result. That does not make it automatically better. It represents a different assumption about how the variations combine.

Table E–5: Linear and statistical stacks

Method	Assumes	Typical use
Linear stack	Contributors may all act unfavourably	Safety-critical fits, sealing interfaces, press fits, controlled clearances
Statistical stack	Contributors are independent and follow an assumed distribution	Non-critical assemblies, large contributor counts, early feasibility studies

Do not use a statistical stack to justify a fit that must be guaranteed in every produced part.

Worked example: enclosure seam

A two-part enclosure has a nominal seam gap of 0.30 mm. The creator must ensure that the assembled parts do not interfere. The contributors are:

Table E–6: Enclosure seam: contributors

Alias	Value	Contributor
BaseGrowth	0.12 mm	Base dimensional change
LidGrowth	0.10 mm	Lid dimensional change

Alias	Value	Contributor
AlignmentVariation	0.06 mm	Location or assembly variation
SeamMargin	0.30 mm	Nominal design gap

Table E–7: Calculation of the combined variation

Alias	Formula
LinearStack	`=BaseGrowth + LidGrowth + AlignmentVariation`
Square1	`=BaseGrowth^2`
Square2	`=LidGrowth^2`
Square3	`=AlignmentVariation^2`
StatisticalStack	`=sqrt(sum(Square1:Square3))`

The linear stack is 0.28 mm. The statistical stack is approximately 0.17 mm. Check the seam:

```
=SeamMargin > LinearStack
```

Assign this the alias `SeamLinearPass`. For the example values, the seam margin exceeds the linear stack, so the worst-case check passes. A status cell can distinguish the two results:

```
=SeamLinearPass ? <<WORST-CASE SEAM PASS>> : <<REVIEW SEAM INTERFACE>>
```

If the linear check fails but the statistical check passes, do not silently accept the statistical result. Record the assumption, identify the contributors and decide whether the configuration may be released, requires review or must be rejected.

Adding a contributor

A range-based stack should make extension deliberate.

To add a fourth contributor:

1. Add a new contributor row with a meaningful alias, for example InsertVariation.
2. Give it an explicit value or formula.
3. Extend the linear range:
```
=sum(Contribution1:Contribution4)
```
4. Add a corresponding squared value:
```
=Contribution4^2
```
5. Extend the statistical range:
```
=sqrt(sum(Square1:Square4))
```
6. Review whether the permitted stack and validation status remain valid.

Do not add a contributor by editing an existing formula in place. Add a named contributor, extend the range and review the result.

Keep contributors meaningful

Each contributor should represent one identifiable physical or engineering effect. Suitable contributors include:

- dimensional change of a mating part;
- shrinkage or growth of a printed component;
- clearance variation;
- fastener or insert position variation;
- assembly alignment variation;
- surface or seam variation;
- tolerance of a purchased component;
- measurement uncertainty, where relevant.

Avoid contributors such as:

- Tolerance1
- Extra
- Safety
- Margin2
- Unnamed cell references

If a value cannot be explained as a physical contributor, it should not enter the stack.

Table E–8: Stack values and decision values

Value	Meaning
Requested clearance or seam	What the design intends
Contributors	Individual sources of variation
Linear stack	Worst-case combined variation
Statistical stack	Assumption-based combined variation
Permitted stack	Declared functional limit
Pass flag	Whether the calculated stack satisfies the limit
Status	What the model reports to the user or release process

A pass flag does not make a configuration validated. It states only that the declared stack rule is satisfied for the current inputs.

Practical rules

- Put each tolerance contributor in its own named cell.
- Use a contiguous range for the contributors.
- Give every contributor a meaningful alias.
- Keep the linear and statistical stacks as separate results.
- State which stack method governs the release decision.
- Use the linear stack for safety-critical or guaranteed-fit interfaces.
- Use a statistical stack only when its assumptions are appropriate and documented.

© 2026 Magimo

- Extend the range deliberately when adding a contributor.
- Keep the permitted stack separate from the calculated stack.
- Test the stack at nominal, minimum and maximum contributor values.
- Review the stack whenever a material, process, interface or fit rule changes.

> A tolerance stack is not one formula; it is a visible set of engineering contributors and a declared decision about how they combine.

E.3.6 Directional compensation and anisotropic scaling

E.3.6.1 Problem

A part may change differently by direction because of process orientation, material behaviour, machining direction, mould shrinkage, coating or post-processing. Uniform scaling may improve one dimension while degrading another.

E.3.6.2 Trick

Keep nominal and compensated dimensions separate, then apply controlled compensation by direction:

```
ScaleX = 1 + CompensationX

ScaleY = 1 + CompensationY

ScaleZ = 1 + CompensationZ

GeometryLengthX = NominalLengthX * ScaleX

GeometryLengthY = NominalLengthY * ScaleY

GeometryHeightZ = NominalHeightZ * ScaleZ
```

For critical interfaces, compensate the individual feature instead of scaling a complete Body:

```
CompensatedHoleDiameter = NominalHoleDiameter + HoleAllowance

CompensatedSlotWidth = NominalSlotWidth + SlotAllowance
```

E.3.6.3 DfAM example

An FDM process profile may use separate in-plane and build-direction compensation:

```
CompensatedLength = NominalLength * (1 + ShrinkXY)

CompensatedHeight = NominalHeight * (1 + ShrinkZ)
```

Use profile-specific values that are tied to the material, machine, orientation, process settings, geometry family and measurement method.

E.3.6.4 Other uses

- Apply finishing allowance by machining direction
- Compensate internal and external laser-cut profiles for kerf

- Allow for timber movement across and along grain
- Account for paint, plating or coating on mating surfaces
- Apply direction-specific mould-shrink allowances

E.3.6.5 Watch-outs

- Directional compensation is empirical data, not a universal material constant.
- Global scaling can corrupt standard hardware interfaces, mates and nominal dimensions.
- Prefer local feature compensation for holes, slots, fits and other functional interfaces.
- Record the active profile identifier, revision, orientation and evidence basis.
- Recheck all coupled features after changing compensation. A correction that improves one dimension can create a new interference elsewhere.

E.3.7 Geometric checks from driving dimensions

Some geometry checks require a dimension that is not directly available as a driving parameter: a diagonal fit, a corner-to-corner distance, the remaining side of a triangle or the clearance around an angled feature. Calculate such check values from known driving dimensions using trigonometry. Do not read the resulting geometry back into the Spreadsheet or sketch that drives it. Reading a geometry-derived value into its own upstream control layer can create a circular dependency.

E.3.7.1 Check architecture

Keep driving dimensions, calculations, permitted values and status separate.

Table E–9: Check architecture: values and their purpose

Value	Purpose
Driving dimensions	Known upstream inputs, such as feature size, bore diameter or interface angle
Calculated geometry value	A derived dimension, such as a diagonal or remaining side
Permitted value	The functional or manufacturing limit
Status	A separate pass, warning or review result

A geometric function calculates a relationship. A separate named check determines whether that relationship satisfies the design requirement.

```
CalculatedValue = <geometry calculation from driving dimensions>

CheckPass = CalculatedValue <relationship> PermittedValue

Status = CheckPass ? <<OK>> : <<REVIEW GEOMETRY>>
```

E.3.7.2 Right-triangle checks

For a right-triangle relationship, use `hypot()` to calculate a diagonal or hypotenuse from two sides:

```
FeatureDiagonal = hypot(FeatureWidth; FeatureHeight)
```

Use `cath()` only when the hypotenuse and one side are known:

```
RemainingSide = cath(DiagonalLength; KnownSide)
```

Check that the inputs form a valid triangle before using the result. Note that `hypot()` requires non-negative side lengths; `cath()` requires the hypotenuse to be greater than or equal to the known side.

```
TriangleInputPass = DiagonalLength >= KnownSide
```

E.3.7.3 Example: diagonal fit in a circular bore

A rectangular feature must fit within a circular bore. The limiting dimension is not the width or height alone, but the diagonal across the feature.

Table E–10: A rectangular fit in a circular bore

Alias	Example value	Meaning
FeatureWidth	30 mm	Feature width
FeatureHeight	40 mm	Feature height
RadialClearance	0.25 mm	Required radial clearance
BoreDiameter	55 mm	Available bore diameter
FeatureDiagonal	`= hypot(FeatureWidth; FeatureHeight)`	Diagonal across the feature
DiagonalFitPass	`= FeatureDiagonal + 2 * RadialClearance <= BoreDiameter`	Fit check

For the example values, the feature diagonal is 50 mm. With 0.25 mm clearance on each side, the required bore diameter is 50.5 mm. The 55 mm bore passes the check. The factor of two is required because the clearance applies to both sides of the diagonal.

```
DiagonalFitStatus = DiagonalFitPass ? <<DIAGONAL FIT PASS>> :
<<INCREASE BORE or REDUCE FEATURE SIZE>>
```

E.3.7.4 Example: remaining triangle side

If the hypotenuse and one side are known, `cath()` returns the other side.

Alias	Example value	Meaning
DiagonalLength	60 mm	Known hypotenuse
KnownSide	36 mm	Known side
RemainingSide	`= cath(DiagonalLength; KnownSide)`	Remaining side
TriangleInputPass	`= DiagonalLength >= KnownSide`	Input validity check

For the example values, the remaining side is 48 mm.

```
TriangleStatus = TriangleInputPass ? <<VALID TRIANGLE INPUT>> :
<<CHECK TRIANGLE DIMENSIONS>>
```

E.3.7.5 Keep the dependency direction one-way

The checks run one way: driving dimension → derived geometry-driving value → permitted-value check → status of downstream geometry. Do not reverse this flow by reading a dimension from the completed geometry back into the sheet or sketch that drives that geometry. That can create a circular dependency or make the model depend on a result that changes during recompute. If a geometry-derived value is needed for reporting, place it in a separate diagnostics or results layer. It may report the result, but it must not become an input to the geometry that produced it.

E.3.7.6 Practical rules

- Calculate check values from known driving dimensions.
- Use `hypot()` for a diagonal or hypotenuse calculated from two sides.
- Use `cath()` only when the hypotenuse and one side are known.
- Validate triangle inputs before using the calculated result.
- Keep the calculated value, permitted value and pass/fail result separate.
- Apply clearances explicitly; do not assume that a diagonal fit leaves radial space automatically.
- Use explicit angle units where an expression requires an angle.
- Test the check at nominal, minimum and maximum feature dimensions.
- Do not feed a geometry-derived value back into the geometry that produced it.

For the next worked pattern – deriving feature counts, pitch and remaining space within a boundary – see E.3.11.

E.3.8 Conditional logic and Boolean rules

E.3.8.1 What conditional logic does

Arithmetic expressions calculate values. Conditional expressions choose between values or responses. A parametric model may need to:

- use a custom dimension instead of a catalogue value;
- apply reinforcement only when a defined condition is met;
- check whether a part fits in the available build envelope;
- reject an incompatible combination of options;
- distinguish a valid parameter set from one that requires review.

FreeCAD expressions support comparisons, conditional selection and Boolean functions. These can be used in Spreadsheet cell formulas and in supported expression-enabled model properties.

> Use conditional logic to encode an explicit design decision – not to hide an undefined or unsupported configuration.

E.3.8.2 Geometry checks without circular references

Geometry checks are covered in the preceding section, Geometric checks from driving dimensions. Calculate check values from known driving dimensions and keep geometry-derived results downstream of the geometry that produced them. Do not feed a geometry-derived value back into the Spreadsheet or sketch that drives that geometry.

E.3.8.3 Basic syntax

A conditional expression has three parts:

```
condition ? resultIfTrue : resultIfFalse
```

In a Spreadsheet cell, start the expression with =:

```
=UseCustom == 1 ? CustomLength : StandardLength
```

When `UseCustom` is 1, the result is `CustomLength`. Otherwise, the result is `StandardLength`.

In a model-property expression editor, enter the expression without the Spreadsheet's leading =. Use explicit units for physical quantities:

```
UseReinforcement == 1 ? 4mm : 2mm
```

Both results represent lengths. Do not mix a physical quantity and a unitless number where the expression requires compatible units.

FreeCAD uses semicolons between function arguments. This avoids ambiguity with decimal commas.

E.3.8.4 Comparison operators

A comparison tests a relationship.

Table E–11: Comparison operators

Operator	Meaning	Example
==	Equal to	`UseCustom == 1`
!=	Not equal to	`SelectedCount != RequiredCount`
>	Greater than	`Clearance > 0 mm`
>=	Greater than or equal to	`WallActual >= WallMinimum`
<	Less than	`BridgeSpan < BridgeLimit`
<=	Less than or equal to	`PartHeight <= BuildHeight`

Use == for an equality test, not a single =.

For measured or calculated dimensions, a tolerance-based comparison is often more meaningful than exact equality:

```
=abs(ActualPitch - TargetPitch) <= PitchTolerance
```

This asks whether the difference is acceptable, rather than whether two calculated numbers are identical.

E.3.8.5 Boolean logic: AND, OR and NOT

Boolean logic combines or reverses conditions.

Table E–12: Boolean functions

Logic	FreeCAD syntax	Meaning
AND[8]	`and(conditionA; conditionB)`	All conditions must be true
OR[8]	`or(conditionA; conditionB)`	At least one condition must be true
NOT	`not(conditionA)`	Reverses the condition

Table E–13: Truth table: results are 1 (true) or 0 (false)

A	B	and(A; B)	or(A; B)	not(A)
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Use the documented FreeCAD functions.

A small, readable rule is:

```
=and(WallPass; ClearancePass; OuterGeometryPass)
```

It is easier to inspect when each named condition has its own visible cell.

E.3.8.6 Examples

Example 1 – A washer that checks its own geometry

Table E–14: Washer dimensions (example)

Alias	Example value
`InnerDiameter`	5.3 mm
`OuterDiameter`	12 mm
`Thickness`	1.2 mm

The selected engineering profile supplies illustrative values.

8 and() and or() can accept more than two arguments.

Alias	Illustrative value
MinimumRadialWall	2 mm
MinimumThickness	1 mm

These numbers illustrate the logic; they are not universal material or manufacturing limits. Create the following calculated cells. Each row's alias belongs to the cell containing its formula.

Table E–15: Washer: calculated checks

Alias	Spreadsheet formula
RadialWall	=(OuterDiameter - InnerDiameter) / 2
DiameterOrderPass	=OuterDiameter > InnerDiameter
BorePass	=InnerDiameter > 0 mm
RadialWallPass	=RadialWall >= MinimumRadialWall
ThicknessPass	=Thickness >= MinimumThickness
WasherParameterPass	=and(DiameterOrderPass; BorePass; RadialWallPass; ThicknessPass)

For the example values, the radial wall is 3.35 mm and all four checks pass. Now change `OuterDiameter` to 8 mm. The radial wall becomes 1.35 mm. The washer may still generate valid CAD geometry, but `RadialWallPass` becomes 0.

Add a diagnostic cell:

```
=RadialWallPass ? <<OK>> : <<Increase outer diameter or reduce bore>>
```

Rule: Distinguish "the shape can be generated" from "the dimensions satisfy the declared design rules".

Example 2 – Build-envelope rotation

A rectangular part may fit the build plate in either of two orientations. For this example, `PartLength`, `PartWidth` and `PartHeight` are already-resolved design dimensions. `BuildX`, `BuildY` and `BuildZ` are usable build dimensions after the required manufacturing margins have been deducted. First test the two orientations separately.

Table E–16: Build envelope: each orientation tested separately

Alias	Spreadsheet formula
FitsOriginal	=and(PartLength <= BuildX; PartWidth <= BuildY; PartHeight <= BuildZ)
FitsRotated	=and(PartWidth <= BuildX; PartLength <= BuildY; PartHeight <= BuildZ)
OuterGeometryPass	=or(FitsOriginal; FitsRotated)

© 2026 Magimo

Then recommend a rotation:

```
=FitsOriginal ? 0 deg : (FitsRotated ? 90 deg : 0 deg)
```

And provide an independent status:

```
= OuterGeometryPass ? <<FITS RECTANGULAR ENVELOPE>> : <<NO FIT IN TESTED
ORIENTATIONS>>
```

Consider a part measuring 220 × 180 × 80 mm and a usable build envelope of 200 × 230 × 250 mm:

- The original orientation fails because 220 mm exceeds 200 mm.
- The rotated orientation passes because 180 mm fits within 200 mm and 220 mm fits within 230 mm.
- The recommended rotation is 90°.

The final `0 deg` in the rotation formula is only a numerical fallback. When `OuterGeometryPass` is false, it must not be interpreted as an approved orientation.

This example tests two in-plane orientations only. It does not establish support requirements, print quality, mechanical suitability or complete packing feasibility.

> **Rule**: a fallback value keeps the output defined; a separate status determines whether that output may be used.

Example 3 – Requested, available and enabled states

A user requests ventilation in an enclosure, but the chosen variant may not permit it. Do not represent all three decisions with one flag.

Table E–17: Requested, available and enabled flags

Alias	Meaning
`VentRequested`	The user asks for ventilation
`VentAvailable`	The selected variant permits ventilation
`VentEnabled`	Ventilation is actually selected by the rules

Resolve the enabled state:

```
=and(VentRequested; VentAvailable)
```

Detect an incompatible request:

```
=and(VentRequested; not(VentAvailable))
```

Assign the second formula the alias `VentConflict`, then display:

```
=VentConflict ? <<Ventilation unavailable for this variant>> : <<OK>>
```

The three states now remain:

Table E–18: The three states (truth table)

Requested	Available	Enabled	Conflict
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	0

The Spreadsheet decides the intended state. Implement that state through a supported, tested mechanism: feature suppression where appropriate, an alternative topology Body or an explicitly defined active result. Do not assume that hiding a feature removes it from the manufactured result or that every feature or Body provides a universal suppression property. See the relevant sections on feature visibility, suppression and active-result selection.

> Preserve what the user requested, what the product permits and what the model actually implements.

Example 4 – Exactly one mounting method

Some choices are alternatives, not additions. A product may permit screw mounting or adhesive mounting, but require exactly one of them.

With clean 0/1 input flags:

```
=MountScrews != MountAdhesive
```

this implements exclusive OR: true when the flags differ.

The equivalent expression using Boolean functions is:

```
=and(or(MountScrews; MountAdhesive); not(and(MountScrews;
MountAdhesive)))
```

The shorter comparison is preferable when both inputs are guaranteed to be binary. For three mounting methods, separate input validity from selection count:

Table E–19: Mounting method: validity and selection count

Alias	Spreadsheet formula
ScrewFlagPass	=or(MountScrews == 0; MountScrews == 1)
AdhesiveFlagPass	=or(MountAdhesive == 0; MountAdhesive == 1)
MagnetFlagPass	=or(MountMagnets == 0; MountMagnets == 1)
MountFlagsPass	=and(ScrewFlagPass; AdhesiveFlagPass; MagnetFlagPass)
MountSelectionPass	=and(MountFlagsPass; (MountScrews + MountAdhesive + MountMagnets) == 1)

Checking only the sum is insufficient if arbitrary numbers can be entered: invalid values could still add up to 1.

Rule: validate the input domain before interpreting the combination.

E.3.8.7 Combine checks without concealing their meaning

Choose one consistent convention:

- A `...Pass` flag is 1 when the requirement is satisfied.
- A `...Fault` or `...Conflict` flag is 1 when a problem exists.

For pass flags:

```
=and(WallPass; ClearancePass; OuterGeometryPass; ConfigurationPass)
```

For fault flags:

```
=not(or(WallFault; ClearanceFault; OuterGeometryFault;
ConfigurationFault))
```

Do not mix the conventions without an explicit conversion. A diagnostic can report the first unmet requirement:

```
=not(WallPass) ? <<CHECK WALL>> : (not(ClearancePass) ? <<CHECK
CLEARANCE>> : <<PARAMETER CHECKS PASS>>)
```

This reports one issue at a time. Keep the individual checks visible so that simultaneous failures are not concealed.

A status cell is not an automatic export interlock. If export must be blocked, the controlled export workflow or generation script must explicitly check the relevant status. Likewise, passing these formulas does not establish that the configuration has been physically validated.

E.3.8.8 Practical rules

- Give each important condition a meaningful alias.
- Use `and()` when every requirement must pass.
- Use `or()` when any permitted alternative is sufficient.
- Use `not()` to reverse a clearly defined condition.
- Keep physical quantities unit-aware and control flags dimensionless.
- Separate requested values, resolved values and diagnostic status.
- Break long expressions into named intermediate steps.
- Do not use a fallback to disguise an invalid configuration.
- Keep validation readbacks downstream of geometry; avoid feeding them back into the sheets that drive that geometry.
- Test both outcomes of each condition and every relevant threshold.

For a threshold rule, test just below the limit, at the limit and just above it. For combined logic, test each individual failure as well as representative combinations. Conditional logic is most useful when it makes the model's decisions visible, explainable and testable.

© 2026 Magimo

E.3.9 Unit stripping and unit attachment

Sometimes a dimension must be converted into a plain number for a count, lookup key, filename or text message. Divide by the relevant unit to remove it:

```
wallCount = floor(availableLength / 1mm / ribPitch)
```

Conversely, attach a unit to a dimensionless value by multiplying by that unit:

```
adjustedClearance = clearanceFactor * 1mm
```

Use this technique deliberately. Keep the underlying engineering value in its natural unit wherever possible and make the intended unit explicit in the alias, description and parameter schema. Do not strip units merely to make a formula easier to write; a unit mismatch is often useful information.

Unit stripping is a formatting tool, not a design shortcut

Dividing by 1 mm, 1 deg or another unit can be useful when a value must become a count, lookup key or text string. Do not use it to bypass unit checking in engineering calculations. Keep physical values in their natural units wherever possible and make any stripped value's purpose explicit through its alias and description.

E.3.10 Named ratios and intermediate values

Do not bury engineering ratios inside long expressions. Define them as named values with a clear physical or functional meaning. For example:

```
gripAspectRatio = gripLength / gripDiameter

transitionLength = gripDiameter * transitionRatio
```

This is preferable to embedding 0.618 or another unexplained factor directly in a sketch expression. A named ratio can be reviewed, tested, documented and changed without tracing every expression that uses it.

Keep intermediate values short-lived and purposeful. If a value has no engineering meaning, it is usually better to simplify the expression than to create another cell.

E.3.11 Integer counts and leftover span

When a repeating feature must fit within a bounded length, derive the count rather than asking the user to enter it independently.

```
featureCount = floor(availableLength / featurePitch)

leftoverLength = availableLength - featureCount * featurePitch
```

Use `ceil()` where the design requires at least one feature and use `round()` only where symmetric distribution is acceptable. Report the leftover length where it affects appearance, symmetry, load path or manufacturability.

This pattern is useful for vent slots, perforations, ribs, corrugations, connector pockets and modular features. Ensure that the minimum count, minimum pitch and edge margins remain explicit validation rules.

© 2026 Magimo

E.3.12 Lists and indexed selection

FreeCAD expressions can create a list of values and retrieve one value from it by index. This is useful when a controlled discrete selection must choose between a small, fixed set of engineering values. The general form is:

```
values = list(value0; value1; value2)
result = values[index]
```

FreeCAD 1.1.2 accepts an index only directly after a name, so give the list its own aliased cell.

Note that FreeCAD listed values in `list()` start at 0. Therefore `values[1]` from `list(10 mm; 20 mm; 30 mm)` returns 20 mm, which feels somewhat counter-intuitive.

Lists are useful when several related values are selected by the same controlled choice. They keep the alternatives visible in one expression, instead of spreading them across several conditional expressions.

E.3.12.1 When to use list()

Use a list-based selection when:

- The number of alternatives is small and stable.
- Each alternative has a clear engineering meaning.
- The selected index is derived from a controlled Configuration Key or discrete choice.
- The values are naturally parallel, for example one thickness per approved material family.
- The complete set of alternatives can be reviewed and validated.

Do not use `list()` when:

- The number of alternatives is likely to grow.
- Different alternatives require different numbers of associated values.
- The list position becomes the only explanation of what a value means.
- A configuration table would make the permitted choices and their consequences clearer.

> A list is a compact selector, not a substitute for a controlled configuration table.

E.3.12.2 Basic pattern

First, convert a controlled text key into an index:

```
materialIndex =

   materialKey == <<PLA_STD_01>> ? 0 :

   materialKey == <<PETG_STD_01>> ? 1 :

   materialKey == <<PA12_STD_01>> ? 2 :

   -1
```

Then retrieve the required value:

`minimumWallOptions = list(1.2 mm; 1.6 mm; 2.4 mm)` and `minimumWall = minimumWallOptions[materialIndex]`. If `materialKey` is `PETG_STD_01`, then `materialIndex` is 1 and `minimumWall` becomes 1.6 mm.

E.3.12.3 Worked example: nozzle-dependent layer height

A 3D-printable model may permit a small set of nozzle sizes. Each nozzle size has a recommended default layer height. The user selects the nozzle profile; the model derives the default layer height.

Alias	Value	Meaning
nozzleKey	NOZZLE_04	Selected nozzle profile

Index calculation

```
nozzleIndex =

   nozzleKey == <<NOZZLE_04>> ? 0 :

   nozzleKey == <<NOZZLE_06>> ? 1 :

   nozzleKey == <<NOZZLE_08>> ? 2 :

   -1
```

Derived value

```
defaultLayerHeight = list(0.20 mm; 0.30 mm; 0.40 mm)
```

Table E–20: Default layer height per nozzle

Selected nozzle	nozzleIndex	defaultLayerHeight
NOZZLE_04	0	0.20 mm
NOZZLE_06	1	0.30 mm
NOZZLE_08	2	0.40 mm
Unknown key	-1	Error or invalid state

Add a validity check

The model should check `nozzleIndex` before using the result. An unknown nozzle key must not silently select the first or last list entry. Therefore, create a separate status alias:

```
nozzleValid = nozzleIndex >= 0
```

Then create an actionable status message:

```
nozzleStatus = nozzleValid ? <<OK>> : <<INVALID NOZZLE: select
NOZZLE_04, NOZZLE_06 or NOZZLE_08>>
```

Do not use the list result unless `nozzleValid` is true.

Note that a list index of -1 is a controlled failure state, not a valid selection.

E.3.12.4 Selecting several related values

Lists are particularly useful when one choice determines several related values. For example, a nozzle selection may determine both a default layer height and a recommended minimum wall.

```
defaultLayerHeight = list(0.20 mm; 0.30 mm; 0.40 mm)

recommendedMinimumWall = list(1.2 mm; 1.6 mm; 2.0 mm)
```

The same `nozzleIndex` selects corresponding values from both lists, so the lists must stay aligned: the same position always means the same nozzle.

Table E–21: Aligned lists: nozzle, layer height and minimum wall

Index	Nozzle	Layer height	Minimum wall
0	NOZZLE_04	0.20 mm	1.2 mm
1	NOZZLE_06	0.30 mm	1.6 mm
2	NOZZLE_08	0.40 mm	2.0 mm

Note that if a new nozzle is added, every parallel list must be updated in the same position. If the lists begin to diverge in length or meaning, replace them with a configuration table.

E.3.12.5 Practical rules

- Use `list()` only for a small, stable set of alternatives.
- Derive the index from a controlled key, not from a raw cell address.
- Start list indexes at 0.
- Check that the index is valid before using the selected value.
- Keep parallel lists aligned: index 0 must mean the same alternative in every list.
- Prefer a configuration table when the alternatives have several associated engineering values.
- Prefer separate tables or a controlled lookup when the number of alternatives may grow.
- Do not use a list to conceal an unsupported choice; an invalid index should produce a visible failure or warning.

E.3.12.6 Watch-outs

- A negative index is not a valid selection. Use it only as an explicit invalid-state marker.
- Do not rely on position alone. A reader should be able to see which alternative corresponds to each index.
- Do not mix unitless counts, lengths and angles in one list unless the meaning is identical for every entry.
- Do not create long lists that duplicate a configuration table. If the alternatives need descriptions, statuses or evidence references, a table is usually clearer.
- Test every valid index, the default selection and at least one invalid key.

© 2026 Magimo

- Use a list when a controlled choice selects one value from a short, fixed set of parallel alternatives.
- Use a configuration table when the choice selects a complete engineering configuration.

© 2026 Magimo

Appendix F Parametric controls outside the Spreadsheet

F.1 Purpose

The Spreadsheet is the control layer of this guide. Several useful controls live elsewhere in FreeCAD: in the property editor, in feature task panels and in the tree view. See for an overview §3.1.5). Most of them accept an expression, so they can be driven from the same aliases as the rest of the model.

This appendix describes two of them. Each section gives what the control does, when to use it, how to set it up in the GUI, and what to watch out for. The controls are advanced. Use them when the main text points to them, not as a first approach.

F.2 Switching a feature on or off: Suppressed

F.2.1 What it does

Every Part Design feature has a Suppressed property. A suppressed feature passes the shape of the previous feature through unchanged. Bound to an expression, it switches an optional feature on or off from a parameter.

F.2.2 When to use it

Setting a dimension to zero, for instance a hole diameter, seems the obvious way to remove a feature. In practice FreeCAD usually stops with a recompute error: a sketch or a Pad of zero size is invalid. The Suppressed property, available since FreeCAD 1.0, avoids this. Use it for an optional feature that the user may include or leave out, such as a mounting tab, a label recess or an extra socket (§4.6.1).

F.2.3 How to implement it

1. In CMParams, add a derived cell, for example `tab_suppressed = UIParams.include_tab == 1 ? 0 : 1`.
2. Select the optional feature in the Model tree.
3. In the property editor, Data tab, find Suppressed.
4. Right-click Suppressed and choose *Expression…*.
5. Enter `CMParams.tab_suppressed` and confirm. If FreeCAD does not find the label, write it as `<<CMParams>>.tab_suppressed`.
6. Change `include_tab` in UIParams and recompute. The feature switches on and off. FreeCAD shows a suppressed feature in grey, struck through, in the tree; you can then see that the step worked.

F.2.4 Watch-outs

- Visibility is not a substitute: a hidden feature still recomputes and still changes the shape (§4.6.3).
- Place optional features late in the feature tree. A later feature that references a face of a suppressed feature loses its reference.

© 2026 Magimo

- Attach the optional feature's sketch to a datum or a master sketch, not to a face that only exists when the feature is active.
- Test both states, then save, close, reopen and test again.

F.3 A drop-down in a VarSet

F.3.1 What it does

A VarSet is a container for named, typed properties. An enumeration property in a VarSet shows as a drop-down list in the property editor. The user picks one option; expressions read the choice.

F.3.2 When to use it

For a discrete user choice with a few fixed options (§3.5.3), when a full Configuration Table (Appendix C) is more than the choice needs. A drop-down offers only valid options, so a typing error cannot occur.

F.3.3 How to implement it (example)

Suppose we want to offer the user a choice for a fit class: loose, normal or tight.

1. Create a VarSet (command *Add a variable set*, Std_VarSet). In the menu, click the symbol that looks like two curly brackets { }.
2. Rename it, for example UserChoices.
3. Select it, right-click in the property editor and choose *Add property*.
4. Choose type *Enumeration*, name `fit_class`, group `Choices`.
5. Enter the options (loose, normal, tight). `UserChoices.fit_class` gives the position of the selected option, counting from 0.
6. Map the choice to a value in CMParams, for example:
   ```
   fit_clearance = UserChoices.fit_class == 0 ? 0.40 mm :
   (UserChoices.fit_class == 1 ? 0.25 mm : 0.15 mm).
   ```
7. While step 6 works, it is generally better to present the option by name, not by position. `UserChoices.fit_class.String` gives its name as text. This ensures that reordering the options, or adding another one later (e.g. press fit), does not break the expression. The expression in the previous step would read:
   ```
   fit_clearance = UserChoices.fit_class.String == <<loose>> ? 0.40 mm :
   (UserChoices.fit_class.String == <<tight>> ? 0.15 mm : 0.25 mm)
   ```

F.3.4 Watch-outs

- Expressions compare the option names. Renaming an option breaks them, so fix the names before release. An option that the expression does not name falls through to its last value (here normal), so extend the expression when you add an option. Add new options at the end of the list.
- For more than three or four options, or when each option carries several values, use a Configuration Table.
- Put the VarSet with the user choices next to UIParams in the tree, so the user finds both in one place.

© 2026 Magimo