European data space
for smart communities

Deliverable 4.1

Replicable recipes for Digital Twins

WP 4 – Recipe-templates, Cookbook and App store

Authors: Bart de Lathouwer (RTD)
Reviewers: Corné Helmons (RTD)
Delivery date: 28-08-2026
Dissemination level: Public
Type: Report

Funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or of the granting authority. Neither the European Union nor the granting authority can be held responsible for them

Revision History

Author Name, Partner short name	Description	Date
Bart de Lathouwer, RTD	First draft version of the deliverable	31-07-2026
Corné Helmons, RTD	Bridge the gap between DT-aCogo and nLDT project insights. Further detailing of the recipes for the usecases	19-08-2026
Bart de Lathouwer, RTD	Final deliverable: the four views of a recipe, the three pilot recipes and their composition, metadata profiles and machine-readable encodings.	28-08-2026

Abbreviations

Abbreviation	Definition	Abbreviation	Definition
API	Application Programming Interface	MIM	Minimal Interoperability Mechanism
CIM	Common Information Model	nLDT	network of Local Digital Twins
DAG	Directed Acyclic Graph	OASC	Open & Agile Smart Cities and Communities
DCAT	Data Catalog Vocabulary (W3C)	OAuth2	Open Authorization 2.0
DS4SSCC-DEP	Data Space for Smart Communities — Deployment	OGC	Open Geospatial Consortium
DT	Digital Twin	OUP	Open Urban Platform (Clearly.hub)
DT-aCoGo	Digital Twin as a Common Good	PPB	Purple Polar Bear (semantic harmonisation)
EIF	European Interoperability Framework	PPI	Pivotal Point of Interoperability
GMT	Goudappel Mobility Twin	SED	Socio-Economic Data
IAM	Identity and Access Management	SKOS	Simple Knowledge Organization System
IMGeo	Informatiemodel Geografie (NL)	SSO	Single Sign-On
JSON	JavaScript Object Notation	WFS	Web Feature Service
3D Tiles	OGC 3D Tiles streaming format	WMS	Web Map Service

Table of Contents

Executive Summary 7
1 Introduction 9
1.1 Purpose of this document 9
1.2 Place in the project 9
1.3 How to read this deliverable 9
2 From kitchen to city: the recipe analogy 10
2.1 What a recipe is in the kitchen 10
2.2 Mapping onto Digital Twins 10
2.3 Where the analogy ends 10
3 The four viewpoints of a recipe 12
3.1 One card, four readers 12
3.2 Viewpoint 1 — The use case 12
3.3 Viewpoint 2 — The ecosystem in the OUP hub 13
3.4 Viewpoint 3 — The calculation model 13
3.5 Viewpoint 4 — The orchestration 14
3.6 How the four viewpoints fit together 14
4 Anatomy of the recipe document 16
4.1 Identity, ingredients and method 16
4.2 Wait vs await 16
4.3 Abstract role and concrete binding 16
4.4 Scenario recipes and live recipes 17
4.5 Lifecycle, versioning and the run manifest 17
Lifecycle 17
Versioning 17
The run manifest 18
5 The pilot cupboard: OUP, CIM and three providers 19
5.1 The Open Urban Platform as the catalogue 19
5.2 Common Information Model: one shopping list, three dialects 19
5.3 Three providers, three interface classes 20
5.4 The semantic readiness check 20
6 Recipe 1 — Pluvial flooding 22
6.1 Viewpoint 1 — The use case 22
Components 22
6.2 Viewpoint 2 — The ecosystem in the OUP hub 22
6.3 Viewpoint 3 — The calculation model 23

6.4 Viewpoint 4 — The orchestration 24
6.5 Requirements 24
Functional 24
Technical 25
6.6 Semantics and ontologies 25
6.7 Catalogue metadata 26
7 Recipe 2 — Urban heat stress 27
7.1 Viewpoint 1 — The use case 27
Components 27
7.2 Viewpoint 2 — The ecosystem in the OUP hub 28
7.3 Viewpoint 3 — The calculation model 28
7.4 Viewpoint 4 — The orchestration 29
7.5 Requirements 29
Functional 29
Technical 29
7.6 Semantics and ontologies 30
7.7 Catalogue metadata 31
8 Recipe 3 — Mobility 32
8.1 Viewpoint 1 — The use case 32
Components 32
8.2 Viewpoint 2 — The ecosystem in the OUP hub 33
8.3 Viewpoint 3 — The calculation model 34
8.4 Viewpoint 4 — The orchestration 35
8.5 Requirements 35
Functional 35
Technical 36
8.6 Semantics and ontologies 36
8.7 Catalogue metadata 37
9 The combined sustainability recipe 38
9.1 Viewpoint 1 — The use case 38
9.2 Viewpoint 2 — The ecosystem in the OUP hub 38
9.3 Viewpoint 3 — The calculation models 38
9.4 Viewpoint 4 — The orchestration 39
9.5 Cross-city binding 39
10 Metadata for recipes, data, mappings, APIs and processes 40
10.1 Recipe record 40

Identity, discovery and governance 40
Semantic binding 41
Ingredients, processing and outputs 41
10.2 Data service record 42
10.3 Mapping and transformation record 43
Header 43
Per attribute 43
Vocabulary governance 44
10.4 API record 44
10.5 Processing service record 45
11 Cookbook, cook and App Store on top of the OUP 46
11.1 Finding is the same; binding is different 46
11.2 Two bodies of work, one card 46
11.3 Scenario recipes and live recipes 46
11.4 Standards, the Blueprint and the MIMs 47
12 Conclusions and next steps 48
12.1 Conclusions 48
12.2 Next steps for the OUP 49
12.2 Next steps 49
12.3 Sources 50
Annex A — Recipe encodings 51
A.1 Pluvial flooding 51
A.2 Urban heat stress 53
A.3 Mobility 54
Tier B — differences only 55
A.4 Combined sustainability recipe 56
Annex B — Binding record 58
Example — pluvial flooding in Rotterdam 59

Executive Summary

This is Deliverable D4.1 of the DS4SSCC-DEP pilot Digital Twin as a Common Good (DT-aCoGo). It describes how ‘recipes’ make a Digital Twin analysis reusable across cities, and how those recipes are found, retrieved, configured and executed. The three pilot cities are Rotterdam, 's-Hertogenbosch and Tampere. The three analyses the pilot runs are pluvial flooding (Rana, Nelen & Schuurmans), urban heat stress (Tygron) and mobility (Dat.mobility, Goudappel). They share a railway-zone planning question and are staged together in Unreal through the Open Urban Platform, Clearly.hub.

The kitchen metaphor does useful work and is kept. A recipe is not the meal and not the restaurant: it is the portable card. Ingredients are data & metadata, models, parameters and the processes that produce them. The method is the order in which those processes are called. A cookbook holds the cards. A cook, the oven, follows the method. The App Store is the menu: a catalogue. In this pilot the running catalogue is the OUP, not a consumer app store. Finding a recipe is the same kind of search as finding a dataset; binding — which city's twin, which provider endpoint, which credentials — is what differs.

The central contribution of this deliverable is that a recipe is read four different ways by four different readers, and that all four belong to the same named card. A planner reads it as a description of a question and the components needed to answer it. A platform administrator reads it as a configuration of data, tools and visualisation in an OUP hub. A modeller reads it as the configuration of a calculation model. A machine reads it as a directed acyclic graph in which the output of one process becomes the input of another. Making these four viewpoints explicit is what allows the operational reality of DT-aCoGo — hubs, registered datasets, provider APIs and an Unreal client — to meet the orchestration encoding developed with Geonovum in the network of Local Digital Twins, without either side having to pretend to be the other.

Chapters 6 to 9 write the three domain recipes and their composition through all four viewpoints, using the models, endpoints, parameters and mappings the pilot integrated. Chapter 10 specifies the metadata that must travel with a recipe, a data service, a mapping, an API and a processing service. Annex A carries the machine-readable encodings; Annex B carries the binding record that resolves an abstract process role to a concrete city endpoint.

The flagship example is the combined sustainability recipe: flood and heat in parallel, a declared water-depth threshold that closes carriageways, then mobility on the remaining network, plated in Unreal with the time slider. That composition is why a directed graph is needed at all. A single model call would be a bookmark.

Today the cities mostly consume pre-computed scenarios registered in the OUP — before, during and after a development. Live recalculation exists in the provider APIs and is treated here as a second, contract-gated tier of the same recipe, recorded as metadata rather than presented as something the Unreal client already performs. The document states which parts are operational and which are specified, so that a reader can tell the difference.

Conclusions. Recipes are the reusable unit DT-aCoGo set out to produce: discussable with planners, configurable in another city's OUP hub, and executable against open process or feature APIs. Interoperability does not come from installing binaries into one vendor's twin. It comes from the common information model, the OUP catalogue, and process interfaces that stay with the provider.

The next steps are twofold:

- OUP: to publish the four recipes as OUP catalogue records with per-city binding records, to carry units and transformations in the CIM mappings, and to let a fourth city bind the same card to its own endpoints.
- Take the practical implementation of the recipes in this document and make abstractions, continue the separation of concerns and prepare to submit work items to OASC MIM8 as well as the OGC Digital Twin working groups.

1 Introduction

1.1 Purpose of this document

DT-aCoGo's goal is to make digital twinning replicable across European cities by using recipe-templates on an urban data space — the Open Urban Platform. D4.1 is the write-up of that unit of reuse: what a recipe contains, who reads it and for what purpose, how it is catalogued, how a city is configured to run it, how a calculation model is bound to it, and how it is executed.

The document is a pilot deliverable, not a standalone essay or standard. It reports what the project designed and integrated, states plainly what is scenario-based today, and points to D3.1 for the model APIs and CIM mappings and to D2.1 for hub set-up. The kitchen analogy is a teaching tool. The evidence is the appendices of D3.1, the CIM workbook, and the hubs the three cities operate.

1.2 Place in the project

DT-aCoGo is a sixteen-month DS4SSCC-DEP pilot running from 1 May 2025 to 31 August 2026. The domains are climate-adaptive urban design and mobility around railway-zone redevelopment. The chain of deliverables this report sits in is:

- **D2.1** — OUP hubs in the three cities, roles, and the data matrix per use case;
- **D2.2** — lifecycle-based time in the twin: the validFrom and validTo attributes that let analysis and visualisation filter identically;
- **D2.3** — checks and guidance by the Governance and Ethical board;
- **D3.1** — interoperability with sectoral models: CIM mappings in Appendix 1 and API descriptions for Dat.mobility, Rana and Tygron in Appendices 2 to 4;
- **D3.6** — ethical assessment of the three calculation models;
- **D4.1** (this document) — the portable recipe that ties those models into one reusable analysis.

One further body of work appears in these pages and should not be confused with the pilot runtime. The public nLDT repositories — App Store, Cookbook and Cook — are an open, OGC-flavoured encoding of find, get and bake, maintained with Geonovum. They show how a recipe can be serialised and executed as a graph of OGC API Processes. They are not the Unreal plugin and they are not Clearly.hub. Chapter 11 sets out how the two fit together and what each contributes – and how to envolve the recipes further.

1.3 How to read this deliverable

Chapter 2 keeps the kitchen analogy. Chapter 3 introduces the organising idea of the document: the four viewpoints of a recipe. Chapter 4 is the anatomy of the recipe document itself. Chapter 5 is the pilot's own cupboard — the OUP, the common information model and the three providers — and closes with the readiness check that tells a city whether a card can be baked there. Chapters 6 to 8 write the three domain recipes through all four viewpoints. Chapter 9 composes them. Chapter 10 specifies the metadata. Chapter 11 returns to cookbook, cook and App Store, now placed on top of the OUP. Chapter 12 states conclusions and next steps. Annex A carries the encodings and Annex B the binding record.

> **A note on what is operational.** Throughout this document, statements about what the pilot runs today are separated from statements about what the recipe specifies. Chapter 3, Table 3 gives the maturity of each of the four viewpoints. Chapter 4.4 distinguishes scenario recipes, which the three cities operate, from live recipes, which the provider APIs support and which a city with the right contracts can switch on without changing the recipe identifier.

2 From kitchen to city: the recipe analogy

2.1 What a recipe is in the kitchen

A published recipe names the dish, lists ingredients with quantities, and describes a method. It does not ship the eggs and it does not include the oven. Anyone who can follow the steps can cook the same dish in another kitchen, with local ingredients that still match the specification — two eggs, not the eggs in my fridge on Tuesday.

A menu is a collection of such cards. A cook executes. The recipe stays the same; the binding to equipment changes. And a recipe assumes that a gram in one kitchen is a gram in another. When it is not — when a local measure means something slightly different — the dish fails quietly rather than loudly. That is the role the common information model plays in Chapter 5.

2.2 Mapping onto Digital Twins

“What happens to flood, heat and traffic if we redevelop the railway zone?” is the dish. The ingredients are city datasets — trees, buildings, pavement — a rain event, socio-economic deltas, and three sector models. The method is the ordered, and sometimes parallel, calling of processes. The extras are the tools: OUP registration, Unreal, a time slider, a callback. The oven is the Digital Twin environment that runs, or displays, the result.

In the kitchen	In DT-aCoGo	Where it lives
Dish	Railway-zone sustainability question	Recipe title and use-case viewpoint
Ingredients	CIM-mapped data, rain event, SED, study area	ingredients block; OUP datasets
Local measures agreed between kitchens	CIM mappings: units, datatypes, transformations	D3.1 Appendix 1; mapping records
Method and order	Flood and heat in parallel, then closures, then mobility	processing graph
Cookbook / menu	Collection of recipe records	OUP catalogue, and the nLDT cookbook encoding
Cook / oven	Executor bound to a chosen twin	A cook, or Unreal calling OUP resources
Menu board	Find recipes like any other resource	App Store as catalogue: DCAT, OGC API Records
Local market	City hub in Clearly.hub	OUP hubs of Rotterdam, 's-Hertogenbosch, Tampere

Table 1 — The kitchen analogy mapped onto the DT-aCoGo stack.

Two properties of a proper method matter later. Steps are directed: you cannot un-bake a cake. And they must not contain cycles. In computing terms the method is a directed acyclic graph.

2.3 Where the analogy ends

A kitchen recipe rarely calls a second kitchen in another country and waits for a webhook. A Digital Twin recipe does: Rana, Tygron and Dat.mobility live at different organisations and are invoked over HTTP. A cookbook on a shelf does not need a trust framework; a bake that touches a municipal traffic model does — OAuth2 client credentials, hub sharing in Clearly.hub, and logging. And unlike a printed card, the machine-readable one must say how results are plated — 3D Tiles, WMS, GeoJSON,

cameras, time — which is why a companion context document sits beside the graph rather than inside it.

There is one further difference, and it is the reason for the next chapter. A kitchen recipe has a single reader. A Digital Twin recipe has four, and they want different things from the same card.

3 The four viewpoints of a recipe

3.1 One card, four readers

During the pilot the word ‘recipe’ was used for four different things, by four different people, and each of them was right. A project coordinator meant a description of a planning question and the components needed to answer it. A platform administrator meant the configuration of datasets, applications and a viewer in a city hub. A modeller at Nelen & Schuurmans, Tygron or Goudappel meant the configuration of a calculation model against a study area. A developer meant a machine-readable graph in which the output of one process becomes the input of the next.

These are not four competing definitions. They are four viewpoints of one artefact, in the sense that an architectural drawing has a plan, a section, an elevation and a services layout. Each viewpoint is complete for its reader and incomplete on its own. A recipe that only exists as a graph cannot be discussed in a planning meeting. A recipe that only exists as a description cannot be executed. And a recipe that describes a question and a graph but says nothing about how a city is configured cannot be replicated, because the second city will discover the missing preconditions one failure at a time.

> **Why this matters beyond DT-aCoGo.** Making the four viewpoints explicit is what allows two bodies of work to meet. DT-aCoGo delivered views 1, 2 and 3 operationally: three cities configured, three providers integrated, an Unreal client that a planner uses. The nLDT work with Geonovum developed viewpoint 4 as an open encoding: an App Store, a Cookbook and a Cook that find, store and execute a recipe as a graph of OGC API Processes. Presented as rival definitions of ‘recipe’ these look like duplicated effort. Presented as views of one card they are complementary, and the boundary between them is a clean interface rather than a disagreement.

The four viewpoints are introduced below and then used as the structure of Chapters 6 to 9, where each of the three pilot recipes and their composition is written through all four.

3.2 Viewpoint 1 — The use case

The first view answers: what question does this card answer, and what does it take to answer it? It is written for a project coordinator, a planner or an alderman's advisor, and it is the view that makes a recipe findable by someone who does not know which model they need.

It names three classes of component, and it is deliberately generic about all three so that the same card can be read by a city that uses different products:

- **Data.** What must be known about the area — trees, buildings, pavement, terrain, a rain event, a socio-economic delta — expressed as common information model feature types where they exist, and declared as unmapped inputs where they do not.
- **Analysis tools.** What kind of calculation is required — a hydrodynamic surface-water simulation, an urban climate simulation, a traffic assignment — expressed as an abstract role, not as a product name. Rana, Tygron and Dat.mobility fulfil these roles in the pilot; another city may bind another provider.
- **Visualisation.** How the answer is shown — which layers, in which order, over which time window, from which viewpoint. This is what the Web 3D Context Document carries, and it is why plating is kept beside the method rather than inside it.

View 1 also carries the things a non-technical reader needs in order to trust the answer: which planning phase the card belongs to, which roles use it, what licence applies, which ethical assessment covers the model, and whether any personal data is involved.

3.3 Viewpoint 2 — The ecosystem in the OUP hub

The second view answers: what must be true of this city before the card can be baked here? It is written for a hub owner or platform administrator, and it is the view DT-aCoGo spent most of its effort delivering.

A city is ready when the following are in place in its OUP hub. The sequence is the same for all three recipes; only the specific datasets and applications differ.

#	Step	Content
1	Hub and governance	A municipal hub exists in Clearly.hub with an owner, participants and a findability setting. Sub-hubs are used where a provider needs its own workspace.
2	Authentication methods	Provider credentials are registered as hub authentication methods so the platform injects them server-side. No secret ever reaches the viewer or the recipe document.
3	Source data registered	The city's own trees, buildings, pavement and supporting datasets are registered as datasets with resources pointing at services at source.
4	Semantic harmonisation	Source attributes are mapped to the common information model, and the harmonised services are registered alongside the originals. Chapter 5.2.
5	Lifecycle attributes applied	validFrom and validTo are set so that analysis and visualisation filter the same objects for the same moment in time, per D2.2.
6	Analysis applications connected	The provider applications are registered, shared to the hub, and given subscriptions or entitlements where the service is gated.
7	Digital twin defined	Layer sets, ordering, visibility and camera presets are saved so the result can be plated consistently.
8	Users and roles assigned	Project engineers and coordinators are invited and given roles, so that actions are attributable.
9	Readiness verified	The semantic readiness check of Chapter 5.4 is run and its result recorded against the recipe.

Table 2 — View 2: the ecosystem configuration a city completes once, then reuses for every recipe.

Steps 1 to 8 are a one-off cost per city, not per recipe. That is the economic argument for the whole approach: the second recipe in a configured hub is far cheaper than the first, and the fourth city inherits a sequence rather than a research project.

3.4 Viewpoint 3 — The calculation model

The third view answers: how is the model itself set up so that it can answer this question for this area? It is written for a modeller, and it is the view that stays with the provider.

This view is deliberately thin in the recipe card and thick in the provider's own documentation, because the pilot found that the three platforms differ sharply in how much a recipe can legitimately dictate. Rana is adaptable and accepts a schematisation path with rainfall parameters. Tygron has a fixed calculation core with a flexible extension layer, so a recipe supplies a location and a tile size and no more. Dat.mobility is configured largely by provider consultants, and its scenarios are prepared in advance.

What the recipe carries in view 3 is therefore not the model configuration itself but the contract with it: which schematisation or project is addressed, which parameters are exposed and with what units,

defaults and ranges, what the execution mode and job lifecycle are, how long results live, and which limitations the provider has declared. Chapter 10.5 specifies these fields.

Provider	What the recipe may set	What stays with the provider
Rana	Schematisation path, rainfall intensity and duration, simulation duration, tile interval, output locations.	The schematisation itself: terrain, drainage network, roughness and boundary conditions.
Tygron	Study location and tile size.	The calculation core, the climate scenario and the vegetation and surface libraries.
Dat.mobility	Study area, phase, and the filters on the returned flows; with an expert-mode contract, a socio-economic delta and network closures.	The underlying traffic model and its maintenance.

Table 3 — View 3: the boundary between recipe and model configuration, as found in the pilot.

3.5 Viewpoint 4 — The orchestration

The fourth view answers: what is called, in what order, with which bindings, and what happens when something fails? It is written for a machine, and it is the view the nLDT encoding was built to express.

A recipe in this view is a document with identity fields a catalogue already understands, ingredients block whose named values are substituted at bake time, and a processing array holding a graph of nodes. Each node has an identifier, a link to a process or a feature collection, inputs that may be literals, ingredient references or references to another node's output, and an execution mode. Dependencies are extracted from those references rather than maintained separately, so that independent nodes in the same wave run in parallel. Chapter 4 sets out the anatomy; Annex A gives the four encodings.

The property that makes this view worth having is composition: the output of one process becomes the input of another. A flood simulation produces water depth; a derivation turns depth above a declared threshold into closed carriageways; a traffic model consumes those closures. No single provider offers that chain, and no single vendor twin needs to own it.

3.6 How the four viewpoints fit together

The views are strata rather than alternatives. Each depends on the ones before it, and each is complete for its own reader.

Viewpoint	Question	Reader	Where it is recorded	Status in this pilot
1 — Use case	What question, and what components?	Project coordinator, planner	Recipe record, catalogue entry; §6.1, 7.1, 8.1, 9.1	Delivered for three use cases and their composition
2 — Ecosystem	Is this city ready?	Hub owner, platform administrator	Hub configuration; §3.3 and §6.2, 7.2, 8.2	Delivered in three cities
3 — Calculation model	How is the model set up for this area?	Modeller at the provider	Provider project plus the recipe's process contract; §6.3, 7.3, 8.3	Delivered for three models

Viewpoint	Question	Reader	Where it is recorded	Status in this pilot
4 — Orchestration	What is called, in what order?	A cook, or the Unreal client	The recipe document; Annex A	Specified; scenario execution operational, live composition not yet the Unreal default

Table 4 — The four views, their readers and their maturity in DT-aCoGo.

Reading the table downwards gives the replication route for a fourth city: understand the question, configure the hub, agree the model contract with a provider, then bake. Reading it upwards gives the assurance route: a graph that executed can be traced back to a model contract, to a configured hub, to a stated question — which is what makes a result defensible in a planning process.

The views also explain a practical fact about DT-aCoGo that is easy to mistake for a shortcoming. The Unreal client collapses views 2, 3 and 4 into a single user action: the planner selects a hub and a simulation, and the client resolves the pilot area, the timeline state and the registered endpoint. That is good product design and poor documentation, because everything that was decided in advance becomes invisible. Writing the views out separately makes the invisible decisions inspectable, and it is what a second city needs to make the same decisions differently.

4 Anatomy of the recipe document

4.1 Identity, ingredients and method

A recipe is a JSON document, or an equivalent serialisation, with identity fields a catalogue already understands: identifier, title, description, version, licence, attribution and links. If it looks like a dataset at the metadata level, the OUP can treat it like a dataset. The difference is not how you find it. The difference is how you bind and bake it.

The ingredients object is the shopping list. Named values — a number, a GeoJSON area, a dataset reference, a process reference — are substituted at bake time. The request that starts the bake may override them, which is how the same Rotterdam method is parameterised for a Tampere railway zone without forking the card. Every ingredient carries a type, and where it is a quantity it carries a unit, a datatype and where useful a default and a permitted range. A number without a unit is the most common way for a portable recipe to fail silently.

Some ingredients are not technical settings but policy choices: the design storm a city plans for, or the water depth above which a street counts as impassable. These are flagged as policy parameters so that they can be approved, versioned and audited rather than buried in a script.

The processing array holds the graph. Each node has an identifier, a link, inputs and an execution mode, and declares what it produces. Terminal outputs are registered back into the OUP as datasets, which is what allows one recipe's result to become another recipe's ingredient.

4.2 Wait vs await

Two node behaviours occur in this pilot, and a recipe must be able to express both. Treating everything as a long-running process would misdescribe the mobility path; treating everything as a simple fetch would misdescribe the other two.

- **Wait, synchronous.** Call and wait for the response. Dat.mobility's segments collection is this kind of call: an OGC API Features fetch of pre-computed flows, not a long job.
- **Await, asynchronous.** Submit, receive an acknowledgement and a job identifier, then poll or await (pub/sub) for a callback. Rana jobs and the Tygron heat process are this kind of call. The node must declare its poll interval, its terminal states, an absolute timeout and what happens when that timeout is reached.

Keeping the Web 3D Context Document beside the graph — layers, cameras, time window — is the same separation as keeping the photograph of the omelette off the list of eggs. Unreal consumes the context; the compute providers can ignore it.

Note: a publish/subscribe method is preferred over polling: better use feedback and not stalling a process execution path (yet more complex to implement)

4.3 Abstract role and concrete binding

Every process node names an abstract role, such as pluvial-flood-simulation, and not a vendor endpoint. The endpoint lives in a separate binding record, one per city, which resolves the role to a registered OUP application, an authentication method, a tenant and a project. Annex B gives the structure.

This separation is what makes the pilot's replicability claim inspectable rather than merely asserted. Rotterdam's binding points at its own Rana tenant; Tampere's points at another; the card is byte-for-byte the same. It is also what protects a published recipe when an endpoint moves or a provider is replaced: the binding record changes and is re-versioned, while the recipe identifier that users have bookmarked does not.

4.4 Scenario recipes and live recipes

At present the three cities consume pre-computed scenarios. Providers do not dynamically pull OUP data to calculate; they use preset, optimised configurations mapped to specific endpoints, and the Unreal client selects among them by pilot area and timeline state. For a sixteen-month pilot that is a defensible engineering choice, and it must be visible rather than implied.

Tier	Name	What the recipe does	Who can bake it
A	Scenario	Retrieval nodes fetch a pre-computed result for a city, domain and time slice — before, during or after. No model run is triggered.	Any hub to which the scenario datasets have been shared. This is what the three cities operate.
B	Live	The full graph: the model is invoked with caller-supplied parameters, the job is awaited, and outputs are registered.	Hubs holding the necessary provider agreements. Supported by the Rana and Tygron APIs today; contract-gated for Dat.mobility.

Table 5 — The two tiers of the same recipe.

The tier is a metadata field, not a second encoding. A city that acquires the necessary agreement switches tier without changing the recipe identifier, and its users keep the same card.

4.5 Lifecycle, versioning and the run manifest

A recipe that informs a planning decision must still be explicable months later, after a provider session has expired and a dataset has been updated. Three mechanisms make that possible.

Lifecycle

State	Meaning	Transition and approving role
draft	Authored; visible only in the owning hub.	Created by a project engineer.
semantic-check	Bindings declared; readiness being verified per city.	The Chapter 5.4 check passes for at least one city.
published	Discoverable and runnable in the listed cities.	Approved by the hub owner; licence chain resolvable; ethical assessment referenced; policy parameters approved.
deprecated	Still resolvable and reproducible; no longer recommended.	Superseded by a newer version, or a binding has degraded.
retired	Retained for provenance; execution disabled.	Binding permanently unavailable, or withdrawn by the Governance and Ethical board.

Table 6 — Recipe lifecycle.

Versioning

- Identifiers are stable and permanent; versions are semantic.
- A change to the graph, to a required ingredient or to an output contract is a major version. Defaults, documentation and non-required metadata are minor or patch changes.
- A published version is immutable. Corrections are issued as new versions, never as edits in place.
- A binding change does not change the recipe version, because the abstract role is unchanged. Binding records carry their own version.

The run manifest

Every execution emits a manifest, registered alongside the result.

Element	Content
runId, startedAt, endedAt	Unique identifier and UTC times.
recipeId, recipeVersion, tier	Exactly which card was baked, in which tier.
actor, hub, role	OUP user, active hub and role at execution time.
ingredientValues	All resolved values, including defaults actually applied.
policyParameters	The policy-bearing subset, with approval references — the design storm, the flood-depth threshold.
bindings	Per node: resolved endpoint, provider, API version, and the OUP dataset and resource identifiers used.
datasetRevisions	Identifier and revision or retrieval timestamp of every CIM service consumed, plus the mapping version in force.
semanticReadiness	The Chapter 5.4 result at run time, including any accepted warnings.
providerJobRefs	Rana job and file descriptor identifiers; the Tygron job identifier and project name.
outputs	Registered dataset and resource identifiers, formats and expiry.
status, diagnostics	Success, partial or failure, with per-node status and messages.

Table 7 — The run manifest.

5 The pilot cupboard: OUP, CIM and three providers

5.1 The Open Urban Platform as the catalogue

In DT-aCoGo the catalogue that cities already operate is Clearly.hub, the Open Urban Platform. Datasets, applications and — in due course — recipes are registered per municipal hub, shared with named counterpart hubs, and protected with the hub's identity.

Each provider is wired in a way that suits its own architecture. Dat.mobility registers three time-tagged scenarios per city plus a technical interface dataset. Tygron registers a heat-calculation dataset per hub and lets Clearly.hub inject the authentication token, so the secret never sits in Unreal. Rana is wired as a multi-tenant API: each municipality has its own organisation, and an API key is configured once per OUP hub.

That is the App Store in production terms: not an installer of binaries, but a catalogue of descriptions pointing at endpoints. The platform already supplies most of what a recipe needs — identity, hubs, dataset and resource identifiers, a credential vault, subscriptions and entitlement, layer configuration, and sharing between hubs. What a recipe adds is a record type of its own, a semantic binding to the common information model, and a run manifest.

5.2 Common Information Model: one shopping list, three dialects

D3.1 Appendix 1 is the workbook that maps local source attributes onto a common information model for trees, buildings and pavement. It is the ingredient list written so that a recipe can ask for cim:Trees2d/treeHeight rather than BOOMHOOGTE in 's-Hertogenbosch, HOOGTE in Rotterdam or Pituusluokka in Tampere. Without that sheet a recipe is a Dutch or a Finnish script, not a European card.

CIM attribute	's-Hertogenbosch	Rotterdam	Tampere	Definition anchored to
cim:Trees2d/treeHeight	BOOMHOOGTE	HOOGTE	Pituusluokka	Environmental thesaurus
cim:Trees2d/treeDiameter	DIAMETER	DIAMETER	Rungon_ymparys	Environmental thesaurus
cim:Trees2d/scientificName	LATBOOMSOORT	BOOMSORTIMENT	Kasvilaji	Darwin Core
cim:Buildings2d/identifier	identificatie	identificatie	PYSYVA_RAKENNUSTUNNUS	—
cim:Buildings2d/intendedUse	gebruiksdoel	gebruiksdoel	RAKENNUSLUOKKA	INSPIRE current use
cim:Pavement2d/function	functie	ONDERDEELTYPE, GEBRUIKSFUNCTIE, WEGTYPE	(value-level: asfaltti, betoni, kiveys …)	IMGeo road function
cim:Pavement2d/terrain	—	VERHARDINGSGROEP	(value-level)	IMGeo road appearance
validFrom, validTo	n/a — set on registration	n/a — set on registration	n/a — set on registration	INSPIRE base

Table 8 — The same CIM ingredients in three city dialects. Source attribute names are given; where a city maps at value level rather than attribute level this is stated.

Four honest limitations travel with this workbook, and they belong in the recipe metadata rather than in a footnote.

- **Units.** Tampere's Rungon_ymparys is a trunk girth, mapped to treeDiameter. Circumference and diameter differ by a factor of π, and heat and wind models care. The mapping must carry the transformation, not only the target field name.
- **Datatypes.** Tampere's Pituusluokka is a height class, mapped to treeHeight, which elsewhere holds a numeric height. A class is not a measurement, and a recipe that averages it will produce a plausible and wrong answer.
- **Precedence.** Rotterdam maps three source attributes onto the single attribute function. Where they disagree, the mapping must say which wins.
- **Vocabulary sovereignty.** Several pavement and road codes are anchored on the Dutch IMGeo catalogue even on the Tampere sheet. That works inside this pilot and does not scale; Finnish values should resolve to Finnish national registers, aligned to the CIM rather than translated through a third country's standard.

Extending the workbook with datatype, unit, transformation, precedence and confidence columns is the smallest change that makes the portability claim testable, and it is the first of the next steps in Chapter 12.

Three inputs the models actually need — terrain, drainage network and socio-economic totals — are not CIM feature types today. The recipes below mark them as unmapped inputs rather than pretending the CIM already covers them.

5.3 Three providers, three interface classes

Provider	Dish	Interface	What Unreal consumes today
Rana (Nelen & Schuurmans)	Constant-rain flood on a schematisation	Multi-step REST job: tenant, project, process execution, polling, 3D Tiles and WMS	3D Tiles and WMS of a completed run, registered in the OUP
Tygron	Heat overlay on a 500 by 500 metre tile	OGC API Processes style at /dtacogoheat with jobs and outputs; conformance partial	WMS overlay; the provider session lives about fourteen days
Dat.mobility (Goudappel)	Traffic flow on road segments	OGC API Features at the segments collection; the Goudappel Mobility Twin also offers an OGC API Processes interface	Three fixed scenarios — before, during, after — time-tagged in the OUP

Table 9 — The three model integrations as they actually behave.

A recipe node must therefore be allowed to be a feature retrieval as well as a process execution. Treating everything as OGC API Processes would misdescribe Dat.mobility's operational path and would hide a commercial gate — expert-mode recalculation — behind a friendly verb.

The three platforms also differ in how much a recipe may configure, which is the practical limit on portability. Rana is adaptable. Tygron has a fixed core with a flexible extension layer. Dat.mobility is configured largely by provider consultants. This is recorded per process in the metadata of Chapter 10.5 so that a fourth city can see what it is taking on before it binds.

5.4 The semantic readiness check

Whether a card can be baked in a given city is a question with a different answer per city and per recipe, and it should be answered by a check rather than by assertion. The check runs when a recipe is offered to a city, and again at execution, with its result recorded in the run manifest. It is the

artefact that turns ‘replicable’ into something testable, and it completes the validation step of the ecosystem configuration in view 2.

#	Check	Method	Failure meaning
1	Feature type availability	Every required CIM feature type has a registered harmonised service in the city hub.	not-ready — the city must complete its mapping.
2	Attribute coverage	Every required attribute appears in the applicable mapping record.	partial — may run with documented substitutions.
3	Unit and datatype agreement	Mapping target unit and datatype match the ingredient contract.	not-ready — the check that catches a girth used as a diameter, or a class used as a measurement.
4	Codelist alignment quality	Value alignments are exact or close matches, not broad or unmapped.	partial, with a comparability warning.
5	Temporal consistency	The required scenario slices exist with validFrom and validTo.	partial — some scenarios unavailable.
6	Coordinate reference system	Service CRS is supported or reprojectable without unacceptable loss.	partial — a reprojection step is required.
7	Binding availability	Every node role has a binding whose endpoint status is healthy.	not-ready — the binding must be repaired or replaced.
8	Entitlement	The hub holds the subscriptions and agreements required.	blocked — the commercial gate is visible before the run, not during it.
9	Governance	Ethical assessment referenced, approval present, policy parameters approved, data-protection obligations satisfied.	blocked — escalate to the Governance and Ethical board.

Table 10 — The nine-step semantic readiness check.

Applied to the pilot as it stands, the check returns ‘partial’ for most combinations and ‘not-ready’ for one. That is the useful result: it names the shortest path to genuine cross-city replicability instead of claiming it is already complete.

Recipe	Rotterdam	's-Hertogenbosch	Tampere
Pluvial flooding	partial — terrain and drainage unmapped	partial — same	partial — same
Urban heat stress	partial — building height and meteorology outside the CIM	partial — same	not-ready — tree attribute unit and datatype defects, check 3
Mobility	partial — socio-economic schema outside the CIM; tier B gated	partial — same	partial — same

Table 11 — Readiness of the three recipes in the three pilot cities.

6 Recipe 1 — Pluvial flooding

Chapters 6 to 9 present each recipe through the four views of Chapter 3, followed by its requirements, its semantic commitments and its catalogue metadata. The machine-readable encodings are in Annex A.

6.1 Viewpoint 1 — The use case

A municipality plans a redevelopment — Merwehaven in Rotterdam, the Spoorzone in 's-Hertogenbosch — and must see where water accumulates during an intense rainfall event, how deep, for how long, and whether the proposed design is better or worse than today. The question is not whether flooding occurs in the abstract but where, at street level, and what changes if the design changes.

Objective	How the recipe delivers it
Quantify pluvial flood risk at street level	Water depth and extent for a declared design storm, at a resolution that distinguishes individual streets and parcels.
Compare current and future states	The same storm run against current-situation and scenario schematisations, filtered consistently by lifecycle attributes.
Treat the design storm as a decision, not a default	Rainfall intensity and duration are policy parameters with units and ranges, set by the city rather than fixed in provider configuration.
Produce defensible evidence	Registered outputs and a run manifest recording ingredients, model version and dataset revisions.
Enable composition	Flood depth is registered as an ordinary OUP resource, so the mobility recipe can consume it as an ingredient.

Table 12 — Objectives of the pluvial flooding use case.

Components

Component class	What is needed
Data	Buildings and pavement from the common information model; a terrain model and a drainage network, which are not CIM feature types today and are declared as unmapped inputs; optionally future-state buildings supplied as 3D Tiles with lifecycle attributes.
Analysis tool	A hydrodynamic surface-water simulation capable of applying a constant rainfall over a schematisation and reporting depth over time. In this pilot the role is filled by Rana.
Visualisation	A time-stepped 3D depth surface for the Unreal twin and a 2D extent map for the web viewer, staged over the development timeline.

Table 13 — Components of the pluvial flooding use case.

6.2 Viewpoint 2 — The ecosystem in the OUP hub

Beyond the generic configuration of Chapter 3.3, this recipe requires the following in the city hub.

Element	Registration	Notes
Harmonised buildings	Dataset with a resource serving cim:Buildings2d	Footprints obstruct and redirect surface flow.
Harmonised pavement	Dataset with a resource serving cim:Pavement2d	Permeability proxy. The richest value-level mapping in the workbook, and where a precedence rule is needed in Rotterdam.

Element	Registration	Notes
Terrain and drainage	Registered where available; declared as unmapped inputs	Required by the model but held inside the provider schematisation rather than harmonised. This is the main limit on portability today and is stated in the recipe rather than hidden.
Future-state buildings	3D Tiles resources with validFrom and validTo	Produced from the BIM pipeline described in D2.1.
Rana application	Registered application with a hub authentication method holding the tenant API key	The key is configured once per hub. Its scope is recorded in the API metadata.
Flood model project	Provider project discoverable by dataset tag	Tenant and project identifiers are read from tags rather than hard-coded, so the card stays city-independent.
Digital twin definition	Layer set including depth tiles, extent map, 3D buildings and trees	Gives the context document something to reference.

Table 14 — Hub configuration for the pluvial flooding recipe.

6.3 Viewpoint 3 — The calculation model

Rana is a multi-tenant data-driven water-analysis workspace. Each DT-aCoGo municipality has its own organisation. Flood models live in the project at a fixed path, and results from Unreal runs are written to a timestamped results folder as a results archive and a 3D Tiles archive.

The process is simple to explain to a planner and strict to encode. A constant rainfall is applied over the whole schematisation for a rain duration, after which the simulation continues dry so that retention, runoff and drainage can be watched. The method is physics- and mathematics-based rather than trained: conservation of mass and fluid dynamics, including groundwater flow and open-surface-water equations. It therefore needs less data than a learned model and its outputs are easier to verify. It is suited to surface-water systems on land and in cities, and explicitly less suited to oceans, deep groundwater and drinking-water networks; that limitation travels in the process metadata.

Parameter	Unit	Default	Notes
schematisation	Rana path	—	For example dt-acogo/floodmodels/Goudsesingel huidig
rain_intensity	mm per hour	80	Policy parameter: the design storm the city plans for
rain_duration	minutes	60	Policy parameter; must not exceed the simulation duration
simulation_duration	minutes	120	Total simulated period including the dry phase
tile_interval	minutes	5	Time step of the 3D Tiles output

Table 15 — The model contract: what the recipe may set.

Everything else — terrain, drainage, roughness, boundary conditions — stays in the schematisation and is the modeller's responsibility. Of the three platforms Rana is the most adaptable, which is why it is the natural first candidate for a tier B live recipe.

6.4 Viewpoint 4 — The orchestration

The method, abbreviated from the Unreal call sequence in D3.1 Appendix 3, is already a graph. The full encoding is in Annex A.1.

Node	Mode	Behaviour
discover	sync	Select the hub; find Rana datasets by tag; read tenant and project identifiers from tags.
submit	async	Process execution with the JSON body of inputs and output paths.
await	poll	Poll the job about every five seconds until completed or failed, with a declared absolute timeout.
tiles_3d	sync	Enumerate and retrieve the 3D tilesets, one per time step.
map_wms	sync, optional	Obtain the map descriptor for the 2D extent map.
register	sync	Publish the results as OUP datasets with lifecycle attributes, so that other recipes can treat flood depth as an ordinary ingredient.
cleanup	sync, optional	Delete the transient provider working directory once registration has succeeded.

Table 16 — The pluvial flooding graph.

Every node names an abstract role — pluvial-flood-simulation — and resolves its endpoint through a binding record. Rotterdam's binding points at its own tenant; Tampere's points at another. The card does not change.

6.5 Requirements

Functional

ID	Requirement
FR-1.1	Accept a design storm defined by rainfall intensity and duration within a simulated period, enforcing that rain duration does not exceed simulation duration.
FR-1.2	Select a schematisation corresponding to a named pilot area and scenario state.
FR-1.3	Execute asynchronously, poll to a terminal state and surface progress to the caller.
FR-1.4	Return results in both a 2D map form and a 3D streaming form, for the web viewer and the Unreal twin respectively.
FR-1.5	Return a time series rather than a single maximum-extent snapshot, since retention and runoff are only observable over time.
FR-1.6	Register terminal outputs as OUP datasets with lifecycle attributes.
FR-1.7	Declare CIM dependencies and unmapped inputs, and refuse to run where readiness fails on unit or datatype agreement.
FR-1.8	Clean up transient provider-side working directories after successful registration.
FR-1.9	Record the design storm as a policy parameter in the run manifest.

Technical

ID	Requirement
TR-1.1	Invoke over HTTPS through the OUP proxy path for the registered provider resource. No credential appears in the recipe document.
TR-1.2	Authenticate with a machine-to-machine API key held in the hub authentication method, with credential scope recorded in the API metadata.
TR-1.3	Discover tenant and project identifiers from OUP dataset tags rather than hard-coding them.
TR-1.4	Declare poll interval, terminal states, an absolute timeout and the behaviour on timeout.
TR-1.5	Retrieve results through descriptor lookup and tileset enumeration, with defined behaviour when a step fails independently.
TR-1.6	Declare the coordinate reference system per city and make any reprojection explicit.
TR-1.7	Register outputs with explicit retention; treat provider working folders as transient.
TR-1.8	Capture provider job and file descriptor identifiers in the run manifest.

6.6 Semantics and ontologies

Concept	Vocabulary	Status
Buildings	cim:Buildings2d; intended use anchored to the INSPIRE current-use codelist	In use
Paving and surfaces	cim:Pavement2d; function and appearance anchored to the IMGeo catalogue	In use; see the sovereignty limitation in Chapter 5.2
Object lifecycle	validFrom and validTo, INSPIRE base types	In use
Terrain and elevation	INSPIRE Elevation; proposed cim:TerrainModel	Unmapped input; on the CIM roadmap
Drainage network	The national urban-water data dictionary, aligned to INSPIRE utility networks; proposed cim:DrainageNetwork	Unmapped input; on the CIM roadmap
Flood results	Proposed cim:FloodResult carrying water depth, flow velocity, extent and time step	Required so that results are consumable rather than opaque tiles
Hazard framing	INSPIRE Natural Risk Zones	Proposed, to give the output a recognised European framing
Precipitation	CF conventions and WMO terminology	Proposed, for when the design storm is replaced by observed or projected series
Provenance	PROV-O	Used by the run manifest

Table 17 — Semantic commitments of the pluvial flooding recipe.

6.7 Catalogue metadata

Field	Value
identifier, version, tier	dt-acogo:recipe/pluvial-flood · 1.0.0 · tier B, with a tier A scenario variant
title	Pluvial flood simulation for area redevelopment
decisionQuestion	Where does water accumulate during a design storm, and does the proposed development improve or worsen it?
projectPhase, userRole	Preparation and Initiative; project engineer runs, project coordinator interprets
requiresFeatureType	cim:Buildings2d, cim:Pavement2d
requiresAttribute	cim:Buildings2d identifier, validFrom, validTo; cim:Pavement2d function, terrain
unmappedInputs	terrain-model, drainage-network, land-use
semanticReadiness	partial in all three cities — terrain and drainage outside the CIM
policyParameters	rain_intensity, rain_duration
recipeLicence, bindingLicence, dataLicence	CC-BY-4.0; Rana provider terms; per registered dataset
ethicalAssessmentRef	D3.6, assessment of the Rana calculation model
personalDataCategory	none
outputs	flood-depth-timeseries as 3D Tiles per time step; flood-extent-map as WMS; both registered with lifecycle attributes
attribution	Nelen & Schuurmans; Purple Polar Bear; the city as data holder; Future Insight

Table 18 — Recipe record for pluvial flooding.

7 Recipe 2 — Urban heat stress

7.1 Viewpoint 1 — The use case

The same redevelopment changes how heat accumulates at street level. Removing trees, adding built mass and replacing permeable ground with hard surfacing all alter how heat builds and how it is experienced by people walking through the area. The planning question is whether the design meets the city's heat standard on a hot day, and which interventions — keeping mature trees, changing surfacing, adjusting massing — matter most.

Objective	How the recipe delivers it
Assess perceived heat at street level	A named thermal-comfort index across the study tile, rather than an air-temperature average.
Attribute effects to design choices	Vegetation, surface material and building geometry drawn from the CIM where mapped and declared where not.
Compare current and future states	Current and scenario datasets selected by lifecycle attributes, as in the flood recipe.
Deliver results into the twin	A map overlay consumable by both viewers, plus a reference into the provider environment for detailed inspection.
Remain defensible after the session expires	Registration into the OUP and a run manifest, because the provider result is transient.
Be comparable between cities	The index and its assumptions declared on the output feature type.

Table 19 — Objectives of the urban heat stress use case.

Components

Component class	What is needed
Data	Trees, buildings and pavement from the common information model; building height and vegetation volume, which reach the model through 3D Tiles rather than the harmonised layer; meteorological conditions, which sit outside the CIM and are supplied by the provider's climate scenario.
Analysis tool	An urban climate simulation producing a spatial heat indicator from built-environment characteristics and meteorology. In this pilot the role is filled by Tygron.
Visualisation	A heat overlay staged over the 3D city with the development timeline, alongside the flood and mobility layers in the combined view.

Table 20 — Components of the urban heat stress use case.

> **A comparability point worth carrying beyond this pilot.** A heat result is comparable across cities only if the index and its assumptions — the reference person, activity level, clothing and meteorological conditions — are declared. Two cities can both report ‘heat stress’ and mean materially different quantities. Declaring the index on the output feature type is a small change with a large effect, and it is a candidate Pivotal Point of Interoperability in OASC terms. It is not yet in the CIM, and Chapter 12 proposes adding it.

7.2 Viewpoint 2 — The ecosystem in the OUP hub

Element	Registration	Notes
Harmonised trees	Dataset serving cim:Trees2d	Height, trunk dimension and species. The most influential input for shade, and the one carrying the unit and datatype limitations of Chapter 5.2.
Harmonised buildings and pavement	Datasets serving cim:Buildings2d and cim:Pavement2d	Massing, shading and surface material as an albedo and thermal-mass proxy.
3D scenario tiles	3D Tiles resources per scenario	Registered per scenario — current, and one per development variant. These carry the height and volume the 2D mapping omits.
Meteorological data	Registered as an original, unharmonised resource	Outside the CIM; the provider otherwise applies its own climate scenario.
Tygron application	One heat-calculation dataset per hub, with the authentication token held by the platform	Clearly.hub injects the token so that Unreal never stores it.
Digital twin definition	Layer set including the heat overlay	The overlay is a map service, so it composes with the flood and mobility layers.

Table 21 — Hub configuration for the heat stress recipe.

7.3 Viewpoint 3 — The calculation model

Tygron is exposed to DT-aCoGo through a dedicated WebApp Proxy API at webapp.tygron.com/dtacogo, with an OGC-style API at /dtacogo/api/ogc. The process used by the pilot is dtacogoheat. The interface follows the OGC API Processes pattern but does not yet fully conform to it, and that partial conformance is recorded in the API metadata rather than glossed over.

Parameter	Unit	Default	Notes
locationX, locationY	degrees	—	Clicked in the client, or supplied directly
size	metres	500	Produces a 500 by 500 metre study tile

Table 22 — The model contract: what the recipe may set.

The calculation core is fixed and the extension layer is flexible, so the recipe supplies a location and a tile size and no more. Three properties of this integration are declared in the process metadata because a second city inherits them along with the card:

- **Results are transient.** The map service remains valid for the provider session, about fourteen days. A planner who captures a map in March cannot assume the URL still resolves in June, which is why registration into the OUP is part of the graph rather than an optional extra.
- **The deployment is project-scoped.** The WebApp is deployed for the purposes of this project and continued availability beyond it is not guaranteed. The binding record carries a support level of ‘as-is’ and a validity window.
- **Usage is attributed to project credentials.** All calls are attributed to the credentials configured for the project rather than to the acting user or hub. Where an audit trail down to the individual user is required, this must be addressed with the provider; the API metadata records the limitation so that it is visible before it matters.

Loading vegetation directly from the common information model is developed on the provider side but not yet connected. Until it is, vegetation reaches the model through the provider's own libraries, and the recipe says so.

7.4 Viewpoint 4 — The orchestration

Node	Mode	Behaviour
submit	async	Execute the heat process for the study location and tile size.
await	poll	Poll the job until completed or failed, with a declared timeout.
outputs	sync	Read the job outputs: the map service URL and the provider project name.
register	sync	Publish the overlay as an OUP dataset with the index definition, the retention warning and lifecycle attributes.

Table 23 — The heat stress graph. Full encoding in Annex A.2.

7.5 Requirements

Functional

ID	Requirement
FR-2.1	Accept a study location and an area size, defaulting to a 500 metre square tile.
FR-2.2	Execute asynchronously, expose a job identifier and poll to a terminal state.
FR-2.3	Return a map service for the computed indicator plus a provider project reference.
FR-2.4	Declare result transience and register any result intended for reuse or evidence.
FR-2.5	Declare the thermal-comfort index and its assumptions on the output feature type.
FR-2.6	Declare building height and meteorology as unmapped inputs, and consume vegetation from the CIM once the provider supports it.
FR-2.7	Carry no credential in the recipe; rely on platform-side token injection.
FR-2.8	Carry the requesting user and hub context to the provider where the interface permits, and record the limitation where it does not.

Technical

ID	Requirement
TR-2.1	Invoke through the OUP proxy for the registered per-city resource, with the token injected server-side.
TR-2.2	Record the interface class as OGC API Processes with partial conformance and dated evidence, rather than asserting conformance.
TR-2.3	Support both request-body execution and the parameterised execute form.
TR-2.4	Declare terminal states, timeout and failure behaviour for polling.
TR-2.5	Supply the location in geographic coordinates and reconcile the study area against city data held in a national grid.
TR-2.6	Declare output retention of approximately fourteen days; require durable registration for any published result.
TR-2.7	Record the support level as as-is and outside provider support agreements, with a validity window.

ID	Requirement
TR-2.8	Record actor attribution as shared project credential until provider credentials are aligned with end-user accounts.

7.6 Semantics and ontologies

Concept	Vocabulary	Status
Vegetation	cim:Trees2d; height and trunk dimension anchored to an environmental thesaurus, species to Darwin Core	In use, with the unit and datatype limitations of Chapter 5.2
Buildings	cim:Buildings2d; intended use anchored to INSPIRE	In use; height absent from the mapping
Building geometry and height	INSPIRE Buildings or the CityGML building module; proposed cim:Buildings3d	Unmapped input; on the CIM roadmap
Vegetation volume and crown	CityGML vegetation module, or a crown-diameter attribute on cim:Trees2d	Proposed; also the natural home for the girth-to-diameter transformation
Surfaces and materials	cim:Pavement2d anchored to IMGeo	In use
Meteorological variables	CF conventions and WMO terminology for air temperature, wind speed and radiation	Outside the CIM; on the roadmap
Heat-stress indicator	Proposed cim:HeatStressResult carrying the named index, its reference assumptions and the computation method	Proposed; the operationalisation of the comparability point in 7.1
Land cover	INSPIRE Land Cover	Proposed

Table 24 — Semantic commitments of the heat stress recipe.

7.7 Catalogue metadata

Field	Value
identifier, version, tier	dt-acogo:recipe/heat-stress · 1.0.0 · tier B, with a tier A scenario variant
title	Urban heat stress assessment for area redevelopment
decisionQuestion	Does the proposed design keep street-level heat within the city's standard on a hot day, and which interventions help most?
requiresFeatureType	cim:Trees2d, cim:Buildings2d, cim:Pavement2d
requiresAttribute	cim:Trees2d treeHeight, treeDiameter, scientificName; cim:Buildings2d identifier; cim:Pavement2d terrain
unmappedInputs	building-height, meteorological-conditions, land-cover
semanticReadiness	partial in Rotterdam and 's-Hertogenbosch; not-ready in Tampere pending correction of tree attribute units and datatypes
indexDefinition	To be declared: the named thermal-comfort index with its reference assumptions
bindingLicence, supportLevel	Tygron provider terms; as-is, outside support agreements
ethicalAssessmentRef	D3.6, assessment of the Tygron calculation model
outputRetention	About fourteen days at the provider; durable once registered
actorAttribution	shared project credential — declared limitation
personalDataCategory	none
attribution	Tygron; Purple Polar Bear; the city as data holder; Future Insight

Table 25 — Recipe record for urban heat stress.

8 Recipe 3 — Mobility

8.1 Viewpoint 1 — The use case

Redevelopment changes how many people live, work and study in an area, and how they move through it — during construction as well as afterwards. The planning question concerns traffic volumes on individual road segments across the phases of the development, and how the network behaves when part of it is unavailable.

Objective	How the recipe delivers it
Quantify traffic per road segment	Vehicle flow per hour for each segment in the study area.
Represent the development phases	Three registered scenarios per municipality — before, during and after — selected by lifecycle attributes so the time slider picks the right one.
Relate mobility to land-use change	A socio-economic delta for the area: homes, resident and job forecasts, commercial floor space and public parking.
Account for conditions	Flow filtered by weather, temperature, mode, date and time, and trip purpose.
Support cross-domain analysis	Accept flood-derived closures as an optional ingredient, which is what makes the combined recipe of Chapter 9 possible.
Make the commercial boundary explicit	Tier A is entitled by dataset sharing; tier B requires a provider agreement. Both are declared, so a city knows before it binds.

Table 26 — Objectives of the mobility use case.

Components

Component class	What is needed
Data	The road network, represented today through cim:Pavement2d; a socio-economic development profile, which is provider-specific and declared as an unmapped input; a study area polygon; and optionally flood-derived closures.
Analysis tool	A traffic assignment capable of returning flow per segment for a given area, phase and set of conditions. In this pilot the role is filled by the Goudappel Mobility Twin over Dat.mobility.
Visualisation	Segment flows styled over the road network, staged with the development timeline.

Table 27 — Components of the mobility use case.

8.2 Viewpoint 2 — The ecosystem in the OUP hub

Element	Registration	Notes
Road segments	Dataset serving cim:Pavement2d	Registered in Rotterdam as a common-model resource. Pavement polygons carry no network topology, which is the semantic weakness noted in 8.6.
Mobility key figures	Registered as both an original and a common-model resource	A clear worked example of the harmonisation pattern: the same content, before and after mapping.
Scenario datasets	Three per municipality, time-tagged with validFrom and validTo	Before, during and after. These are what the Unreal time slider selects between.
Technical interface dataset	One per city	Describes the API resource rather than carrying data.
Dat.mobility application	Registered with an OAuth2 client-credentials authentication method, unique per municipality	The best-scoped credential model of the three providers, and a good reference pattern.
Entitlement	Scenario access by dataset sharing; expert mode by provider agreement	Recorded so that the gate is visible in the catalogue rather than discovered at run time.

Table 28 — Hub configuration for the mobility recipe.

8.3 Viewpoint 3 — The calculation model

Dat.mobility wraps municipal traffic models in the Goudappel Mobility Twin. Because the twin can work from several established modelling packages, the recipe binds to an API rather than to a specific engine — architecturally the closest of the three providers to the loose binding this deliverable argues for.

A scenario is built from a study area polygon plus a delta of socio-economic data: new homes, resident and job forecasts, commercial floor space and expected public parking. The provider extends the polygon with influence and boundary zones, so the returned extent is deliberately larger than the requested one and consumers must not assume they are equal. Missing fields are estimated where possible and otherwise ignored; the recipe declares which behaviour applies rather than leaving it to be discovered.

For DT-aCoGo the cities chose the fixed-scenario path. The Processes interface can do more — road closures, socio-economic edits and a recalculated network in the order of ten to sixty seconds — but those functions are not applied here, and expert mode requires an agreement with the provider to maintain the underlying model. Configuration is performed largely by provider consultants, which is the practical limit on self-service recipes for this domain.

Tier	What the recipe does	Interface	Who can bake it
A	Fetch pre-computed segment flows for before, during or after, with time, weather, mode and purpose as filters.	OGC API Features at the segments collection	Any hub to which the scenario datasets have been shared
B	Submit an area and a socio-economic profile, optionally with closures; await recalculation; then fetch flows.	OGC API Processes	Holders of a model-maintenance agreement; not the default user

Table 29 — The two tiers of the mobility recipe.

8.4 Viewpoint 4 — The orchestration

The operational path is a single synchronous retrieval, which is exactly why a recipe must permit feature retrieval as well as process execution. Encoding this node as a process would misdescribe what happens and would hide the commercial gate behind a friendly verb.

Node	Mode	Behaviour
fetch_flow (tier A)	sync	Retrieve items from the segments collection for the study area, phase and conditions; each feature carries a flow attribute in vehicles per hour.
apply_profile (tier B)	async	Submit the socio-economic profile and any closures; await recalculation, typically ten to sixty seconds.
fetch_flow (tier B)	sync	Retrieve the recalculated flows once the modification has been applied.

Table 30 — The mobility graph. Full encoding in Annex A.3.

> **An open semantic question, recorded rather than smoothed over.** The provider states that the socio-economic JSON complies with the common data model published by Purple Polar Bear, while the CIM workbook currently documents only trees, buildings and pavement. A recipe cannot bind to an ambiguity. For version 1.0.0 the block is therefore declared a provider-specific ingredient, and a CIM roadmap item is opened for a documented cim:AreaDevelopmentProfile feature type. Choosing explicitly is what matters; the choice can be revised in a later version without changing the recipe identifier.

8.5 Requirements

Functional

ID	Requirement
FR-3.1	Retrieve traffic flow per road segment for a defined study area.
FR-3.2	Select a development phase from the registered scenario set using lifecycle attributes.
FR-3.3	Permit weather, temperature, mode, date and time, and trip purpose as caller-supplied filters.
FR-3.4	Declare the tier and its entitlement condition, defaulting to tier A.
FR-3.5	Where entitled, support runtime modification of the socio-economic profile and of network availability.
FR-3.6	Accept an optional flood-derived closure list.
FR-3.7	Declare the personal-data character of the socio-economic inputs.
FR-3.8	Record the node as a feature retrieval where the segments collection is used.
FR-3.9	Declare deterministic behaviour for absent socio-economic fields.

Technical

ID	Requirement
TR-3.1	Authenticate with OAuth2 client credentials provisioned per municipality.
TR-3.2	Supply the study area as a valid polygon and do not assume the returned extent equals the requested one.
TR-3.3	Supply socio-economic input in the documented structure with declared null handling.
TR-3.4	Record the interface class as feature retrieval for the operational path and as process execution for the entitled path.
TR-3.5	Use synchronous execution with a short timeout for scenario retrieval, and asynchronous handling for entitled recalculation.
TR-3.6	Return results as geospatial features carrying a flow attribute per segment.
TR-3.7	Check entitlement before execution and report clearly where expert mode is unavailable.

8.6 Semantics and ontologies

Concept	Vocabulary	Status
Road segments	cim:Pavement2d with function anchored to IMGeo	In use — a weak proxy for a transport network, since pavement polygons carry no connectivity
Transport network	INSPIRE Transport Networks, or the European transport model vocabulary; proposed cim:RoadSegment with topology	Proposed; required before flows can be reliably joined across cities
Traffic flow results	Proposed cim:TrafficFlowResult with flow, unit, temporal reference and mode; alignable with the European traffic data exchange standard	Proposed; makes the output reusable rather than provider-specific
Socio-economic demand	Proposed cim:AreaDevelopmentProfile; a statistical exchange vocabulary for the forecast ranges	Roadmap item opened in 8.4
Trip purpose and mode	Codelists published as SKOS concept schemes	Proposed; currently free-text values
Temporal reference	Lifecycle attributes plus explicit date-time semantics for the modelled hour	Partially in use
Disruption input	cim:FloodResult from Chapter 6	Proposed

Table 31 — Semantic commitments of the mobility recipe.

8.7 Catalogue metadata

Field	Value
identifier, version, tier	dt-acogo:recipe/mobility-scenario · 1.0.0 · tier A operational; tier B variant entitlement-gated
title	Traffic flow per development phase
decisionQuestion	How does traffic on each road segment change before, during and after the development, and what happens when the network is disrupted?
requiresFeatureType	cim:Pavement2d
unmappedInputs	area-development-profile (provider-specific in 1.0.0), road-network-topology
semanticReadiness	partial in all three cities — socio-economic schema outside the CIM; network represented as pavement
requiresSubscription	Tier A: scenario datasets shared to the hub. Tier B: provider model-maintenance agreement
bindingLicence	Goudappel provider terms; expert mode contractually gated
ethicalAssessmentRef	D3.6, assessment of the dat.Mobility calculation model
personalDataCategory, lawfulBasis	aggregate-statistical — resident, job and student forecasts by area; no individual-level data
attribution	Goudappel; Purple Polar Bear; the city as data holder; Future Insight

Table 32 — Recipe record for mobility.

9 The combined sustainability recipe

9.1 Viewpoint 1 — The use case

The proposition of the pilot is not three separate maps. It is a combined view of how a railway-zone redevelopment performs on water, heat and traffic, now and later, so that trade-offs can be discussed before concrete is poured. Keeping a row of mature trees may cool a street and narrow a carriageway. Raising a plinth may keep water out of a building and push it into a neighbouring square. These interactions are the substance of the planning conversation, and no single model produces them.

This is also the only reason a directed graph earns its keep. One model invocation needs no cookbook — a bookmark would do. Composition is what requires ingredients to be named consistently, outputs to be registered as inputs, parallel waves to be identified, and a policy threshold to be declared rather than buried.

Component class	What is needed
Data	Everything the three domain recipes require, plus one derived dataset: the set of carriageways rendered impassable by flooding.
Analysis tools	The three domain roles, plus a small derivation step that intersects flood depth with the road network.
Visualisation	One staged scene: flood depth over time, heat overlay, segment flows, 3D buildings and trees, with a single timeline and camera.

Table 33 — Components of the combined use case.

9.2 Viewpoint 2 — The ecosystem in the OUP hub

The combined recipe adds no new registration requirement beyond the union of the three domain recipes, with one addition: the derived closure dataset must be registrable, so that it can be passed as an ingredient and retained as evidence. This is a useful property of the design — a city that has configured its hub for the three domain recipes is already configured for the composition.

9.3 Viewpoint 3 — The calculation models

The three model contracts of Chapters 6.3, 7.3 and 8.3 apply unchanged. The composition introduces one contract of its own: the derivation step, which is not a provider model but a small, transparent operation owned by the project.

Aspect	Content
Operation	Intersect the flood depth surface with the road network and mark as impassable those segments where depth exceeds a declared threshold.
Threshold	A policy parameter with a unit, a default of 0.15 metres, an approval reference and a version. It is not permitted to hide in a script.
Why it is a policy parameter	Deciding when a road counts as impassable is a judgement with consequences for the traffic result and therefore for the planning advice. A governance board can approve it; another city can set it differently and say so.
Ownership	The project, not a provider — which is precisely why it must be documented as carefully as the provider models.

Table 34 — The derivation contract.

9.4 Viewpoint 4 — The orchestration

The graph has two waves and a dependent chain. Flood and heat are independent of one another and run in parallel; mobility depends on the closures derived from the flood result.

Node	Role	Depends on	Behaviour
flood	Sub-recipe (Chapter 6)	—	Asynchronous. Wave 1.
heat	Sub-recipe (Chapter 7)	—	Asynchronous, independent of flood. Wave 1, in parallel.
closures	Derivation	flood	Intersects depth above the declared threshold with the road network.
mobility	Sub-recipe (Chapter 8)	closures	Tier A selects the matching scenario; tier B passes the closure list to the live interface.
register	Publication	flood, heat, mobility	Registers terminal outputs as OUP datasets and writes the run manifest.
stage	Presentation	register	Emits the context document: layers, camera and time window.

Table 35 — The combined graph. Full encoding in Annex A.4.

In today's Unreal path the during-construction scenario already encodes known works, and the flood-driven closure is the research step rather than routine operation. The graph records the intended composition so that the step from one to the other is a configuration change rather than a redesign.

9.5 Cross-city binding

Cross-city binding is the same card and three hubs. Rotterdam, 's-Hertogenbosch and Tampere each supply locally mapped trees, buildings and pavement, a local flood schematisation, a local heat-calculation registration and a local scenario set. The method does not fork; the binding records do. Whether a given city can bake the card today, and what is missing if it cannot, is answered by the readiness check of Chapter 5.4 rather than by assertion — and the answer for all three cities is currently ‘partial’, for the reasons given in Table 11.

> **What this demonstrates and what it does not.** The pilot demonstrates that a single card can describe an analysis that three cities configure and run against their own data and their own provider endpoints. It does not demonstrate that the full live composition runs unattended today: the three cities operate tier A, and the flood-driven closure step is research. Stating both plainly is what makes the first claim credible.

10 Metadata for recipes, data, mappings, APIs and processes

A recipe travels only as far as its metadata allows. This chapter specifies the five record types the pilot found necessary. Each is designed to remain a valid catalogue record for a harvester that knows nothing about recipes: DT-aCoGo terms sit in their own namespace alongside standard DCAT-AP fields, so that search stays on OGC API Records and the extension is additive.

Three principles govern the whole profile:

- **Extend, do not replace.** A recipe record is a catalogue record first. Finding a recipe should not require new infrastructure; only binding and baking differ.
- **Separate the abstract role from the concrete binding.** The recipe names what must be done; the binding record names who does it in this city. This is what lets an endpoint move without reissuing the card.
- **Make constraints visible rather than abstracting them away.** Commercial gating, transient results, shared credentials and unmapped inputs must all be expressible. A card that hides a constraint exports risk to the next city that adopts it.

Record type	Describes	Basis
Recipe record	The portable analytical card.	DCAT-AP core plus a DT-aCoGo extension; OGC API Records for discovery.
Binding record	How an abstract role resolves in one city.	DT-aCoGo; see Annex B.
Data service record	A service delivering data at source.	dcat:DataService, GeoDCAT-AP, the OUP dataset and resource model.
Mapping record	A source-to-CIM schema and value transformation.	DT-aCoGo; PROV-O for lineage, SKOS for codelist alignment.
Processing service record	An executable model and the API through which it is called.	OGC API Processes descriptions plus governance and assurance fields.

Table 36 — The five record types.

In the tables below, M means mandatory, R recommended and O optional.

10.1 Recipe record

Identity, discovery and governance

Field	Ob.	Type or example	Notes
identifier	M	dt-acogo:recipe/pluvial-flood	Stable, resolvable, never reused.
version	M	1.0.0	Semantic. Published versions are immutable.
title, description	M	Multilingual literal	Written for view 1: a planner must recognise the question.
owningHub, findability	M	Hub identifier; PUBLIC, LIMITED or PRIVATE	Mirrors the OUP visibility model.
status	M	draft … retired	Lifecycle per Chapter 4.5.
tier	M	A-scenario or B-live	Metadata, not a second encoding.
projectPhase, userRole	R	Preparation, Initiative; project engineer, coordinator	Aligns the card with the municipal planning cycle.

Field	Ob.	Type or example	Notes
decisionQuestion	R	Free text	The planning question answered — what makes a card findable by a non-technical reader.
recipeLicence	M	CC-BY-4.0	Applies to the recipe document only.
bindingLicence, dataLicence	M	Provider terms; per dataset	Three distinct concepts. Conflating them is how an openly licensed card acquires hidden obligations.
requiresSubscription	R	Entitlement reference	Where commercial gating exists, it is visible before the run.
ethicalAssessmentRef	R	Reference into D3.6	Connects each bound model to its published assessment.
personalDataCategory, lawfulBasis, dpiaReference	R	none, aggregate-statistical or personal	Required for the mobility recipe.
approvedBy, approvalDate	M when published	Hub owner; the Governance and Ethical board where policy parameters exist	Uses the project's own role model.
attribution	M	Ordered provider list	One canonical list per recipe.

Semantic binding

Field	Ob.	Type or example	Notes
cimVersion	M	cim:1.0	The CIM carries a resolvable namespace and a version.
requiresFeatureType	M	cim:Buildings2d, cim:Pavement2d	The explicit dependency on the harmonised layer.
requiresAttribute	M	cim:Trees2d/treeHeight	Attribute level, because readiness fails at attribute level.
unmappedInputs	M	terrain-model, drainage-network	Inputs the model needs that the CIM does not yet cover. Declared, not hidden.
semanticReadiness	M	Per city: ready, partial or not-ready, with reasons	Computed by the Chapter 5.4 check.
conformsToProfile	R	dt-acogo:profile/pluvial-flood/1.0	The abstract contract a bound process must satisfy.

Ingredients, processing and outputs

Field	Ob.	Type or example	Notes
ingredient.name, type	M	rain_intensity; scalar, dataset-ref, process-ref, geometry, structured	Every ingredient is typed.
ingredient.unit, datatype	M	mm/hour, decimal	A quantity without a unit is how a portable recipe fails silently.

Field	Ob.	Type or example	Notes
ingredient.default, range	R	80; 0 to 200	Defaults are recorded so that a run can be reproduced.
ingredient.policyParameter	R	true for the design storm and the depth threshold	Marks a judgement rather than a technical setting.
ingredient.cimBinding	R	cim:Pavement2d/terrain	Links a parameter to the harmonised model.
ingredient.source	R	OUP dataset and resource identifiers	Resolves to entitlement, not to a bare URL.
node.role	M	pluvial-flood-simulation	Abstract; resolved through the binding record.
node.interfaceClass	M	ogcapi-processes, ogcapi-features, rest-job, wms	A node may be a feature retrieval as well as a process execution.
node.execution	M	sync or async; poll interval, terminal states, timeout, behaviour on timeout	Failure semantics are part of the contract.
node dependencies	M	References such as :submit.outputs.jobId	Extracted from references rather than maintained separately.
output.role, format, register, retention	M	flood-depth-timeseries; 3D Tiles; register as dataset; provider expiry	Terminal outputs become catalogued resources, which is what enables composition.

10.2 Data service record

A data service is what an ingredient resolves to. The OUP already models this as a dataset with resources; the fields below are what a recipe needs in order to bind safely and what the readiness check needs in order to evaluate.

Field	Example	Why the recipe needs it
serviceType	WMS, WFS, OGC API Features, 3D Tiles, SensorThings	Determines which node interface class may consume it.
endpointURL, endpointDescription	Service URL plus capabilities or OpenAPI document	Machine-resolvable capability discovery.
servesFeatureType	cim:Buildings2d	The join to the CIM. Without it, readiness cannot be computed.
appliedMapping	Mapping record identifier and version	Distinguishes an original city service from a harmonised one — the distinction the Rotterdam registrations already make.
crs, resolution, accuracy	National grid; positional accuracy	The pilot cities use two different national grids.
temporalCoverage, validFrom, validTo	Scenario time window	Implements the lifecycle rule that drives the time slider.
scenarioRef	current, before, during, after	A recipe must be able to select a registered scenario.
updateFrequency, lastVerified	Monthly; date of last successful fetch	Prevents silent staleness.

Field	Example	Why the recipe needs it
accessRights, licence, subscriptionRef	Open, restricted or paid	Enforced by the hub.
authenticationMethod	Hub authentication method reference	Keeps secrets server-side.
personalDataCategory	none, aggregate or personal	Required for socio-economic inputs.

Table 37 — Data service metadata.

10.3 Mapping and transformation record

This is where the portability claim is won or lost. A recipe that asks for a tree diameter must be able to discover that one city supplies a diameter, another a girth, and a third a height class — and what to do about it. The workbook is therefore part of the recipe, not an annex for data specialists.

Header

Field	Content
mappingId, version, validFrom	Mappings evolve independently of both data and recipes and are versioned separately.
city, sourceDataset, sourceService	What is harmonised, by OUP identifier.
targetFeatureType, cimVersion	cim:Trees2d at cim:1.0.
executedBy, method	Purple Polar Bear; schema mapping, semantic alignment and transformation.
provenance	PROV-O: agent, activity and the harmonised service generated.
validationStatus, validatedBy, validationDate	Makes the workbook auditable rather than merely present.
knownLimitations	Surfaced to recipe consumers — for example a diameter derived from a girth, or a height supplied as a class.

Per attribute

Field	Example (Tampere)	Example (Rotterdam)	Purpose
sourceAttribute	Rungon_ymparys	DIAMETER	Origin attribute.
targetAttribute	cim:Trees2d/treeDiameter	cim:Trees2d/treeDiameter	Destination.
definitionURI	Environmental thesaurus concept	Same concept	Semantic anchor.
sourceDatatype, sourceUnit	decimal; centimetres, circumference	decimal; centimetres, diameter	Exposes the difference before it becomes an error.
targetUnit	centimetres, diameter	centimetres, diameter	The target contract.
transformation	divide by π	identity	The correction itself, carried with the mapping.

Field	Example (Tampere)	Example (Rotterdam)	Purpose
precedence	—	1 of 3 sources for function	Where several source attributes map to one target, which wins.
codelistAlignment	SKOS exact, close or broad match	—	Value-level quality, needed for pavement codes.
confidence	medium	high	Propagates into the readiness result.
nullHandling, defaultValue	omit or default	—	Deterministic behaviour on missing data.

Table 38 — Mapping metadata, per attribute.

Vocabulary governance

- The CIM carries a resolvable namespace and a version, so that a term such as cim:Trees2d/treeHeight dereferences to a definition with its datatype and unit.
- Codelists are published as SKOS concept schemes with explicit alignments. Finnish values align to Finnish national registers rather than defaulting to another country's standard.
- Feature semantics are separated from geometric representation, so that 3D and temporal variants do not require new type names.

10.4 API record

An API record describes a callable interface independently of the recipe that binds it, so that conformance and security are assessed once and reused.

Field	Example	Notes
apiId, provider, version	Provider-versioned identifier	Not project-versioned.
specification	OpenAPI document URL	Rana publishes OpenAPI in JSON and YAML with rendered documentation; the Mobility Twin publishes a specification.
interfaceClass	ogcapi-processes, ogcapi-features, rest-job	Records what the interface actually is.
conformsTo, conformanceEvidence	Conformance class identifiers; date and result of a check	Separates claimed from verified conformance. The Tygron proxy is recorded as partial.
authenticationType, credentialScope	OAuth2 client credentials per city; API key per tenant; token per hub	Makes over-broad credentials visible.
actorAttribution	end-user or shared-project-credential	Records where the audit trail is complete and where it is not.
secretHandling	proxy-injected, client-supplied, in-URL	Only proxy injection is acceptable for a published recipe.
rateLimits, slaClass	Requests per minute; supported or as-is	Feeds capacity planning and failure semantics.
deprecationDate, successorApi	—	Enables orderly migration when an endpoint is retired.

Table 39 — API metadata.

10.5 Processing service record

Extends the process description with the governance and assurance fields a municipality needs before acting on a result.

Field	Example	Notes
processId, title	dtacogoheat	As published by the provider.
processProfile	dt-acogo:profile/heat-stress/1.0	The abstract role, which is what enables substitution by another provider.
inputs, outputs	Name, datatype, unit, range, default, required	The model contract of view 3.
executionMode, jobLifecycle	async; submit, poll, retrieve	As documented per provider in Chapter 5.3.
typicalDuration, timeout	Seconds to minutes	Recalculation in the Mobility Twin is in the order of ten to sixty seconds.
outputRetention	About fourteen days, or durable once registered	The transience of a provider session is a property of the result, not a footnote.
modelType	physics-based, statistical or learned	The flood model is physics- and mathematics-based rather than trained; this matters for transparency.
ethicalAssessmentRef, assessmentDate	Reference into D3.6	Makes assurance discoverable at the point of use.
fitnessForPurpose, knownLimitations	Not suitable for oceans, deep groundwater or drinking-water networks	Prevents misapplication of a portable card in a fourth city.
indexDefinition	The named index and its reference assumptions	Required wherever the output is an index rather than a measured quantity.
configurability	core-fixed, extensible or fully-configurable	The evidence for why binding must stay loose.
commercialConstraint	Expert mode requires a provider agreement	Visible before binding rather than at run time.
failureSemantics	Retry policy, partial-result behaviour, fallback	What the cook does when a step fails.

Table 40 — Processing service metadata.

11 Cookbook, cook and App Store on top of the OUP

11.1 Finding is the same; binding is different

A recipe record is a valid DCAT-AP record first, with DT-aCoGo terms in a separate namespace, so that a harvester which knows nothing about recipes still sees a catalogue entry. Search stays on OGC API Records. What differs is binding: choosing the oven and the provider endpoints. That is how a Rotterdam card is baked in Tampere without a single European Digital Twin product.

11.2 Two bodies of work, one card

The nLDT split — App Store, Cookbook and Cook as three services — is a good separation of concerns for the market: find, store and execute can be procured independently. In this pilot those three jobs collapse onto the OUP, the provider APIs and the Unreal client. Both descriptions are true, provided they are labelled. It would be false to claim that DT-aCoGo built and operates a public App Store, Cookbook and Cook. It is true that the split is the right architecture, and that the nLDT services are a runnable encoding of it.

The four viewpoints of Chapter 3 make the relationship precise, and this is the deliverable's contribution to the wider network of Local Digital Twins.

Viewpoint	Where DT-aCoGo delivered it	Where nLDT expresses it	Interface between them
1 — Use case	Recipe records for three use cases and their composition, written for planners.	Catalogue entry in the App Store.	The same DCAT-AP record. No translation needed.
2 — Ecosystem	Three configured municipal hubs, with registered datasets, harmonised services, applications and roles.	Assumed; the testbed sandbox stands in for a configured city.	The binding record: it is exactly what a configured hub contributes to a bake.
3 — Calculation model	Three integrated providers with documented contracts, parameters and limitations.	Assumed; stubs stand in for real models.	The processing service record of Chapter 10.5.
4 — Orchestration	Specified in this deliverable; scenario execution operational through Unreal.	Implemented as the Cookbook and the Cook.	The recipe document itself, in Annex A.

Table 41 — How the pilot and the open encoding meet across the four views.

Read across the table, the two efforts are complementary rather than duplicative. The pilot contributes what a research sandbox cannot: three real cities, three commercial models with real constraints, and a common information model that has met three municipal data dialects. The open encoding contributes what a pilot cannot: a runnable, vendor-neutral expression of the graph, maintained in public. The binding record and the processing service record are the two joints where they meet.

11.3 Scenario recipes and live recipes

Providers do not currently pull OUP data dynamically to calculate. They use preset, optimised configurations mapped to specific endpoints, and the client selects among them. Tier A, the scenario recipe, is what the three cities operate. Tier B, the live recipe, is what the APIs already support and what a follow-on city with the right agreements can switch on without changing the recipe identifier. The tier is metadata, not a second encoding, and Chapter 4.4 sets out both.

11.4 Standards, the Blueprint and the MIMs

The recipe idea touches four DS4SSCC Blueprint areas: catalogue and discovery, data and service description, identity and trust, and orchestration. The first three are served by existing mechanisms — DCAT-AP and OGC API Records for the catalogue, the OUP hub model for identity and entitlement, and the profiles of Chapter 10 for description.

The fourth exposes a gap. There is no settled Minimal Interoperability Mechanism for workflow orchestration, MIM8 has the ambition and the landing place for future work on orchestration. A recipe is a candidate Pivotal Point of Interoperability between cities and model providers, and framing it that way — rather than only as a JSON file maintained in one country — is what makes it adoptable across Europe. DCAT's discomfort with compute modules and methods is a related question for W3C and OGC; the pilot's contribution is a concrete profile rather than only the diagnosis.

The pilot offers two candidate Pivotal Points of Interoperability to OASC and DS4SSCC:

- **The recipe profile.** Abstract role separated from concrete binding, constraints made visible, existing platform identifiers reused, and a readiness check that says whether a card can be baked in a given city. Chapter 10 and Annexes A and B are the specification.
- **The declared index.** Wherever a model output is an index rather than a measured quantity — heat stress being the clearest case — the index and its reference assumptions must travel with the result. Without this, two cities can report the same indicator and mean different things, and no amount of catalogue interoperability will detect it.

Both are small, testable and independent of any product. Both came out of trying to make one card work in three cities, which is the kind of evidence a Blueprint refinement should rest on.

12 Conclusions and next steps

12.1 Conclusions

- **This specific recipe encoding is the portable unit DT-aCoGo set out to produce:** a named dish, a CIM-aware shopping list, and a directed method. It is neither the data nor the Digital Twin product.
- **One card is read four ways.** As a use case with data, analysis tools and visualisation; as an ecosystem configuration in an OUP hub; as a calculation model contract; and as an orchestration graph. Naming the four viewpoints is what allows the pilot's operational work and the open orchestration encoding to meet without either misrepresenting the other.
- **The three dishes the pilot cooks are real:** Rana flood, Tygron heat and Dat.mobility flow. Their APIs, parameters, registrations and limitations are documented in D3.1 and are carried here on one card each, plus a composition.
- **The CIM workbook is part of the recipe.** City dialects of tree height and pavement function must resolve to the same ingredient names, with units, datatypes, transformations and precedence spelled out. This is where portability is genuinely won.
- **Composition is the reason for the graph.** Flood depth above a declared threshold closes carriageways, which changes traffic. A single model call would need no cookbook.
- **Policy parameters must be declared, not buried.** The design storm a city plans for and the depth at which a street counts as impassable are judgements with consequences. They are ingredients, they carry units and approval references, and a governance board can approve them.
- **Today the bake is mostly tier A.** The APIs already offer tier B. The document records both, so that a reader is not sold a live orchestration the client does not yet perform.
- **Replicability is testable, not assumed.** The readiness check returns ‘partial’ for most city and recipe combinations today. That is a more useful statement than a claim of completeness, because it names the remaining work.

12.2 Next steps for the OUP

#	Action	Why it matters
1	Publish the four recipes as OUP catalogue records, with binding records per city hub.	Makes the cards findable and bindable rather than only described.
2	Extend the CIM workbook with datatype, unit, transformation, precedence and confidence; correct the Tampere girth and height-class mappings; align Finnish pavement values to Finnish registers.	The smallest change that makes cross-city comparability testable.
3	Add CIM feature types — or maintain explicit unmapped inputs — for terrain, drainage network, road network topology and the area development profile.	These are the four inputs that currently limit portability.
4	Declare the heat index and the flood-depth closure threshold as versioned, approved ingredients.	Turns two hidden judgements into governed decisions.
5	Emit a run manifest for every bake and register terminal outputs with lineage.	Lets a planning decision be defended months later, after provider sessions have expired.
6	Run the readiness check as a gate before a recipe is offered in a city, and publish the result.	Replaces an assertion of replicability with a check.
7	Complete the provider-onboarding section of the OUP governance documentation from view 1 and view 2 content.	Gives a fourth city and a fourth provider a written route in.
8	Keep the nLDT services as the open encoding of find, get and bake, clearly labelled as such alongside the operational OUP and Unreal path.	Preserves the complementarity set out in Chapter 11.2.

Table 42 — Next steps.

12.2 Next steps

#	Action	Why it matters
1	Abstract, profile and extend the OUP specific recipe concept and its viewpoints for future standardization in OASC (MIM8) and further in OGC (Digital Twin working groups)	Create finer grained, more abstract specifications to fit in the current and future OGC API Processes specifications (multiple parts)
2	Offer the recipe profile to DS4SSCC as a candidate orchestration MIM, and the declared index as a second candidate;	Turns pilot findings into Blueprint refinement, which is the pilot's obligation.

12.3 Sources

Internal DT-aCoGo sources first; public encodings and standards second.

- D2.1 — Setting up OUP hubs in the pilot sites: hubs, roles and the use-case data matrix.
- D2.2 — Lifecycle-based time in the twin: the validFrom and validTo attributes.
- D2.3 — Checks and guidance by the Governance and Ethical board.
- D3.1 — Interoperability between the OUP and sectoral models, including Appendix 1 (CIM mappings workbook) and Appendices 2 to 4 (Dat.mobility, Rana and Tygron API descriptions).
- D3.6 — Ethical assessment of the three calculation models.
- DT-aCoGo OUP Governance and Authentication architecture note.
- Clearly.hub — Open Urban Platform product documentation.
- Geonovum nLDT repositories — App Store, Cookbook and Cook: an open encoding of find, get and bake; not the operational OUP and Unreal path.
- OGC API Records, OGC API Processes and OGC API Features; WMS; WFS; OGC 3D Tiles.
- DCAT-AP and its national application profiles; INSPIRE data specifications; CityGML; PROV-O; SKOS; CF conventions.
- OASC Minimal Interoperability Mechanisms and Pivotal Points of Interoperability; the DS4SSCC Blueprint.

Annex A — Recipe encodings

The four recipes of Chapters 6 to 9 in machine-readable form. Placeholders in braces are resolved through the binding record of Annex B. No credential appears in any recipe document: provider secrets are held as hub authentication methods and injected by the platform.

The encoding is deliberately close to the nLDT cookbook serialisation so that the same card can be read by the open cook and by the OUP. Where this deliverable adds fields — semantics, policy parameters, governance — they sit in the DT-aCoGo namespace and are ignorable by a consumer that does not understand them.

Albeit that these encodings are close to the nLDT cookbook recipes, the below recipes are practical and specific implementation that made the recipes work in the DT-aCoGo / OUP environment. Future work is to take these practical and specific recipes and make them more abstract and modular again for future standardisation.

A.1 Pluvial flooding

```
{
  "id": "dt-acogo:recipe/pluvial-flood",
  "version": "1.0.0",
  "tier": "B-live",
  "conformsToProfile": "dt-acogo:profile/pluvial-flood/1.0",
  "title": "Pluvial flood simulation for area redevelopment",
  "description": "Where does water accumulate during a design storm, and does
                  the proposed development improve or worsen it?",
  "owningHub": "dtacogo-rotterdam",
  "findability": "PUBLIC",
  "status": "draft",
  "projectPhase": ["Preparation", "Initiative"],
  "userRole": ["Project Engineer", "Project Coordinator"],

  "semantics": {
    "cimVersion": "cim:1.0",
    "requiresFeatureType": ["cim:Buildings2d", "cim:Pavement2d"],
    "requiresAttribute": [
      "cim:Buildings2d/identifier", "cim:Buildings2d/validFrom",
      "cim:Buildings2d/validTo", "cim:Pavement2d/function",
      "cim:Pavement2d/terrain"],
    "unmappedInputs": ["terrain-model", "drainage-network", "land-use"]
  },

  "ingredients": {
    "study_area":     { "type": "geometry", "format": "GeoJSON Polygon" },
    "scenario_state": { "type": "scalar", "enum": ["current","future"],
                        "default": "current" },
    "schematisation": { "type": "scalar",
                        "example": "dt-acogo/floodmodels/Goudsesingel huidig" },
    "rain_intensity": { "type": "scalar", "unit": "mm/hour",
                        "datatype": "decimal", "default": 80,
                        "range": [0, 200], "policyParameter": true },
    "rain_duration":  { "type": "scalar", "unit": "minute", "default": 60,
                        "constraint": "<= simulation_duration",
                        "policyParameter": true },
    "simulation_duration": { "type": "scalar", "unit": "minute", "default": 120 },
    "tile_interval":  { "type": "scalar", "unit": "minute", "default": 5 },

    "buildings": { "type": "dataset-ref", "featureType": "cim:Buildings2d",
                   "resource": "{hub}/datasets/{buildingsDatasetId}" },
    "pavement":  { "type": "dataset-ref", "featureType": "cim:Pavement2d",
                   "resource": "{hub}/datasets/{pavementDatasetId}" },

    "floodProcess": { "type": "process-ref",
                      "role": "pluvial-flood-simulation",
```

```
                    "interfaceClass": "rest-job",
                    "binding": "dt-acogo:binding/rana-rotterdam" }
},

"processing": [{
  "id": "pluvial-flood",
  "nodes": [
    { "id": "discover",
      "link": "$floodProcess", "operation": "process_list",
      "body": { "inputs": { "tag": "rana-dtacogo-v1" } } },

    { "id": "submit",
      "link": "$floodProcess", "operation": "process_execution",
      "dependsOn": ["discover"],
      "execution": { "mode": "async" },
      "body": { "inputs": {
          "schematisation": { "variable_type": "rana_path",
                              "id": "$schematisation", "ref": "main" },
          "simulation_duration": "$simulation_duration",
          "rain_duration":       "$rain_duration",
          "rain_intensity":      "$rain_intensity",
          "tile_interval":       "$tile_interval" } } },

    { "id": "await",
      "link": "$floodProcess", "operation": "job_retrieve",
      "execution": { "mode": "poll", "intervalSeconds": 5,
                     "terminalStates": ["completed","failed"],
                     "timeoutMinutes": 60,
                     "onTimeout": "fail-and-report" },
      "body": { "inputs": { "job": ":submit.outputs.jobId" } } },

    { "id": "tiles_3d",
      "link": "$floodProcess", "operation": "get_ogc_3d_tilesets",
      "body": { "inputs": { "job": ":await.outputs.jobId" } },
      "produces": { "role": "flood-depth-timeseries",
                    "format": "OGC 3D Tiles", "perTimestep": true } },

    { "id": "map_wms",
      "link": "$floodProcess", "operation": "file_descriptor_wms",
      "optional": true,
      "body": { "inputs": { "job": ":await.outputs.jobId" } },
      "produces": { "role": "flood-extent-map", "format": "OGC WMS" } },

    { "id": "register",
      "link": { "rel": "process", "role": "oup-dataset-registration" },
      "dependsOn": ["tiles_3d"],
      "body": { "inputs": {
          "resources": [":tiles_3d.produces", ":map_wms.produces"],
          "featureType": "cim:FloodResult",
          "validFrom": "$scenarioValidFrom",
          "validTo":   "$scenarioValidTo" } } },

    { "id": "cleanup",
      "link": "$floodProcess", "operation": "project_delete_directory",
      "dependsOn": ["register"], "optional": true }
  ]
}],

"outputs": [
  { "role": "flood-depth-timeseries", "from": ":tiles_3d", "register": true },
  { "role": "flood-extent-map",       "from": ":map_wms",  "register": true }
],

"governance": {
  "recipeLicence": "CC-BY-4.0",
  "ethicalAssessmentRef": "DT-aCoGo:D3.6#rana",
  "personalDataCategory": "none",
```

```
    "attribution": ["Nelen & Schuurmans", "Purple Polar Bear",
                    "City of Rotterdam", "Future Insight"]
  }
}
```

A.2 Urban heat stress

```
{
  "id": "dt-acogo:recipe/heat-stress",
  "version": "1.0.0",
  "tier": "B-live",
  "conformsToProfile": "dt-acogo:profile/heat-stress/1.0",
  "title": "Urban heat stress assessment for area redevelopment",
  "owningHub": "dtacogo-rotterdam",
  "findability": "PUBLIC",

  "semantics": {
    "cimVersion": "cim:1.0",
    "requiresFeatureType": ["cim:Trees2d","cim:Buildings2d","cim:Pavement2d"],
    "requiresAttribute": [
      "cim:Trees2d/treeHeight", "cim:Trees2d/treeDiameter",
      "cim:Trees2d/scientificName", "cim:Buildings2d/identifier",
      "cim:Pavement2d/terrain"],
    "unmappedInputs": ["building-height", "meteorological-conditions"],
    "readinessNote": "Vegetation is supplied from provider libraries;
                      common-model ingest is on the provider roadmap."
  },

  "ingredients": {
    "locationX": { "type": "scalar", "unit": "degree",
                   "datatype": "decimal", "example": 4.470887 },
    "locationY": { "type": "scalar", "unit": "degree",
                   "datatype": "decimal", "example": 51.921965 },
    "size":      { "type": "scalar", "unit": "metre", "default": 500 },
    "scenario_state": { "type": "scalar", "enum": ["current","future"],
                        "default": "current" },
    "indexDefinition": { "type": "scalar",
                         "note": "named thermal-comfort index and its
                                  reference assumptions; travels to the output" },

    "trees":     { "type": "dataset-ref", "featureType": "cim:Trees2d",
                   "resource": "{hub}/datasets/{treesDatasetId}" },
    "buildings": { "type": "dataset-ref", "featureType": "cim:Buildings2d",
                   "resource": "{hub}/datasets/{buildingsDatasetId}" },
    "pavement":  { "type": "dataset-ref", "featureType": "cim:Pavement2d",
                   "resource": "{hub}/datasets/{pavementDatasetId}" },

    "heatProcess": { "type": "process-ref", "role": "heat-stress-simulation",
                     "interfaceClass": "ogcapi-processes",
                     "conformance": "partial",
                     "binding": "dt-acogo:binding/tygron-rotterdam" }
  },

  "processing": [{
    "id": "heat-stress",
    "nodes": [
      { "id": "submit",
        "link": "$heatProcess",
        "operation": "processes/dtacogoheat/execute",
        "execution": { "mode": "async" },
        "body": { "inputs": { "locationX": "$locationX",
                              "locationY": "$locationY",
                              "size": "$size" } } },

      { "id": "await",
        "link": "$heatProcess", "operation": "jobs/{jobId}",
        "execution": { "mode": "poll", "intervalSeconds": 10,
                       "terminalStates": ["completed","failed"],
                       "timeoutMinutes": 30 },
```

```
      "body": { "inputs": { "job": ":submit.outputs.jobId" } } },

    { "id": "outputs",
      "link": "$heatProcess", "operation": "jobs/{jobId}/outputs",
      "body": { "inputs": { "job": ":await.outputs.jobId" } },
      "produces": [
        { "role": "heat-indicator-map",   "format": "OGC WMS" },
        { "role": "provider-project-ref", "format": "text" } ] },

    { "id": "register",
      "link": { "rel": "process", "role": "oup-dataset-registration" },
      "dependsOn": ["outputs"],
      "body": { "inputs": {
          "resources": [":outputs.produces.heat-indicator-map"],
          "featureType": "cim:HeatStressResult",
          "indexDefinition": "$indexDefinition",
          "retentionWarning": "provider session expires after ~14 days",
          "validFrom": "$scenarioValidFrom",
          "validTo":   "$scenarioValidTo" } } }
  ]
 }],

 "outputs": [
   { "role": "heat-indicator-map",   "from": ":outputs", "register": true },
   { "role": "provider-project-ref", "from": ":outputs" }
 ],

 "governance": {
   "recipeLicence": "CC-BY-4.0",
   "ethicalAssessmentRef": "DT-aCoGo:D3.6#tygron",
   "supportLevel": "as-is",
   "actorAttribution": "shared-project-credential",
   "personalDataCategory": "none",
   "attribution": ["Tygron","Purple Polar Bear",
                   "City of Rotterdam","Future Insight"]
 }
}
```

A.3 Mobility

Tier A is the operational path. Tier B shares the identifier and differs only in the fields shown after it.

```
{
 "id": "dt-acogo:recipe/mobility-scenario",
 "version": "1.0.0",
 "tier": "A-scenario",
 "title": "Traffic flow per development phase",
 "owningHub": "dtacogo-rotterdam",
 "findability": "PUBLIC",

 "semantics": {
   "cimVersion": "cim:1.0",
   "requiresFeatureType": ["cim:Pavement2d"],
   "unmappedInputs": ["area-development-profile", "road-network-topology"]
 },

 "ingredients": {
   "study_area": { "type": "geometry", "format": "GeoJSON Polygon",
                   "note": "the provider extends this with influence and
                            boundary zones; the returned extent is larger" },
   "phase":      { "type": "scalar", "enum": ["before","during","after"],
                   "default": "before",
                   "resolvesVia": "validFrom/validTo on registered scenarios" },
   "datetime":   { "type": "scalar", "format": "ISO-8601" },
   "weather":    { "type": "scalar", "enum": ["dry","rain"], "default": "dry" },
   "purpose":    { "type": "scalar", "enum": ["work","school","other"] },

   "segments": { "type": "dataset-ref", "featureType": "cim:Pavement2d",
```

```
                "interfaceClass": "ogcapi-features",
                "collection": "segments",
                "binding": "dt-acogo:binding/datmobility-rotterdam" }
  },

  "processing": [{
    "id": "mobility-scenario",
    "nodes": [
      { "id": "fetch_flow",
        "link": "$segments", "operation": "items",
        "execution": { "mode": "sync", "timeoutSeconds": 60 },
        "body": { "inputs": { "areaId":   "$study_area",
                              "datetime": "$datetime",
                              "weather":  "$weather",
                              "purpose":  "$purpose" } },
        "produces": { "role": "traffic-flow-segments",
                      "format": "application/geo+json",
                      "attribute": "flow", "unit": "vehicles/hour" } }
    ]
  }],

  "outputs": [ { "role": "traffic-flow-segments", "from": ":fetch_flow" } ],

  "governance": {
    "recipeLicence": "CC-BY-4.0",
    "ethicalAssessmentRef": "DT-aCoGo:D3.6#datmobility",
    "personalDataCategory": "aggregate-statistical",
    "requiresSubscription": "scenario-datasets-shared-to-hub"
  }
}
```

Tier B — differences only

```
  "tier": "B-live",

  "ingredients": {
    "development_profile": {
      "type": "structured",
      "featureType": "provider-specific:AreaDevelopmentProfile",
      "cimRoadmap": "cim:AreaDevelopmentProfile (proposed)",
      "fields": {
        "expected_public_parking_spaces": { "datatype": "integer" },
        "new_homes":              { "datatype": "integer" },
        "residents_forecast_min": { "datatype": "integer" },
        "residents_forecast_max": { "datatype": "integer" },
        "commercial_space":       { "datatype": "decimal", "unit": "m2" },
        "jobs_forecast_min":      { "datatype": "integer" },
        "jobs_forecast_max":      { "datatype": "integer" },
        "students_forecast_min":  { "datatype": "integer" },
        "students_forecast_max":  { "datatype": "integer" } },
      "nullHandling": "provider estimates where possible, otherwise ignored" },

    "network_disruption": {
      "type": "dataset-ref", "featureType": "cim:FloodResult",
      "optional": true,
      "note": "closed segments derived from the flood recipe (Annex A.4)" }
  },

  "processing": [{ "id": "mobility-live", "nodes": [
    { "id": "apply_profile", "link": "$mobilityProcess",
      "interfaceClass": "ogcapi-processes",
      "operation": "runtime-modification",
      "execution": { "mode": "async", "expectedSeconds": [10, 60],
                     "timeoutMinutes": 10 },
      "body": { "inputs": { "area": "$study_area",
                            "profile": "$development_profile",
                            "closures": "$network_disruption" } } },
    { "id": "fetch_flow", "link": "$segments", "operation": "items",
```

```
      "dependsOn": ["apply_profile"] } ] }],

  "governance": {
    "requiresSubscription": "provider-model-maintenance-agreement",
    "entitlementNote": "Not available to hubs holding scenario access only."
  }
```

A.4 Combined sustainability recipe

```
{
  "id": "dt-acogo:recipe/combined-sustainability",
  "version": "1.0.0",
  "title": "Combined Sustainability Simulation",
  "description": "Combined assessment of water retention and drainage, urban
                  heat stress and mobility flows for a railway-zone
                  redevelopment, now and after completion.",
  "owningHub": "dtacogo-rotterdam",
  "findability": "PUBLIC",
  "projectPhase": ["Preparation","Initiative"],
  "userRole": ["Project Engineer","Project Coordinator"],

  "composes": [
    { "recipe": "dt-acogo:recipe/pluvial-flood",     "version": "^1.0.0" },
    { "recipe": "dt-acogo:recipe/heat-stress",       "version": "^1.0.0" },
    { "recipe": "dt-acogo:recipe/mobility-scenario", "version": "^1.0.0" }
  ],

  "ingredients": {
    "study_area":     { "type": "geometry" },
    "scenario_state": { "type": "scalar", "enum": ["current","future"] },
    "rain_intensity": { "type": "scalar", "unit": "mm/hour", "default": 80,
                        "policyParameter": true },
    "flood_depth_threshold": {
        "type": "scalar", "unit": "metre", "default": 0.15,
        "policyParameter": true,
        "approvedBy": "{governanceBoardRef}",
        "note": "depth above which a carriageway counts as impassable" }
  },

  "processing": [{
    "id": "combined-sustainability",
    "nodes": [
      { "id": "flood", "recipeRef": "dt-acogo:recipe/pluvial-flood",
        "with": { "rain_intensity": "$rain_intensity",
                  "study_area": "$study_area",
                  "scenario_state": "$scenario_state" },
        "execution": { "mode": "async" } },

      { "id": "heat", "recipeRef": "dt-acogo:recipe/heat-stress",
        "with": { "study_area": "$study_area",
                  "scenario_state": "$scenario_state" },
        "execution": { "mode": "async" } },

      { "id": "closures",
        "link": { "rel": "process", "role": "network-impact-derivation" },
        "dependsOn": ["flood"],
        "body": { "inputs": {
            "flood":     ":flood.outputs.flood-depth-timeseries",
            "network":   "$segments",
            "threshold": "$flood_depth_threshold" } },
        "produces": { "role": "impassable-segments" } },

      { "id": "mobility", "recipeRef": "dt-acogo:recipe/mobility-scenario",
        "dependsOn": ["closures"],
        "with": { "network_disruption": ":closures.produces",
                  "study_area": "$study_area" } },

      { "id": "register",
```

```
        "link": { "rel": "process", "role": "oup-dataset-registration" },
        "dependsOn": ["flood","heat","mobility"],
        "body": { "inputs": { "runManifest": true } } },

      { "id": "stage",
        "link": { "rel": "process", "role": "context-document-emit" },
        "dependsOn": ["register"] }
    ]
  }],

  "presentation": {
    "contextDocument": {
      "layers": ["flood-depth-timeseries","heat-indicator-map",
                 "traffic-flow-segments","buildings-3d","trees-3d"],
      "timeline": { "from": "$scenarioValidFrom", "to": "$scenarioValidTo" },
      "camera": { "preset": "railway-zone-overview" }
    }
  }
}
```

Annex B — Binding record

A binding record resolves one abstract process role, for one recipe, in one city. It is versioned separately from the recipe, so that an endpoint can move, a credential can be rescoped or a provider can be replaced without reissuing the card that users have bookmarked.

Field	Content
bindingId, recipeId, nodeRole, city, version	Which role is resolved, for which recipe, in which city.
providerOrg, appId	The provider organisation and the registered OUP application identifier.
endpoint, apiRecord	The resolved URL — normally the OUP proxy path — and a reference to the API record of Chapter 10.4.
tenantRef, projectRef, discovery	Where identifiers are discovered from dataset tags rather than hard-coded, the tag patterns used.
authenticationMethod, credentialScope, secretHandling	A reference to the hub authentication method, the scope of the credential, and confirmation that the secret is injected server-side.
actorAttribution	Whether calls are attributable to the end user or only to shared project credentials.
validFrom, validUntil, endpointStatus, lastChecked	The availability window and the result of the last conformance check.
supportLevel	supported, best-effort or as-is.
resultRetention	How long a provider-side result remains valid.
entitlement	Any subscription or agreement required to use this binding.

Table 43 — Binding record structure.

Example — pluvial flooding in Rotterdam

```
{
  "bindingId": "dt-acogo:binding/rana-rotterdam",
  "recipeId":  "dt-acogo:recipe/pluvial-flood",
  "nodeRole":  "pluvial-flood-simulation",
  "city": "Rotterdam",
  "version": "1.0.0",
  "providerOrg": "Nelen & Schuurmans",
  "appId": "{oupAppId}",
  "endpoint": "{oupProxy}/active-hub/{hubId}/datasets/{datasetId}
              /resources/{resourceId}/v1/tenants/{tenant}",
  "discovery": { "datasetTag": "rana-dtacogo-v1",
                 "tenantTag":  "rana-tenant-*",
                 "projectTag": "rana-project-*" },
  "authenticationMethod": "{hubAuthMethodId}",
  "credentialScope": "tenant",
  "secretHandling": "proxy-injected",
  "actorAttribution": "end-user",
  "supportLevel": "supported",
  "resultRetention": "provider folder transient; durable once registered",
  "endpointStatus": "healthy",
  "lastChecked": "2026-08-28"
}
```

The bindings for 's-Hertogenbosch and Tampere differ only in city, hub, tenant and dataset identifiers. The equivalent records for the heat and mobility roles differ in provider, authentication type and support level: the heat binding carries a support level of as-is, a result retention of about fourteen days and an actor attribution of shared project credential; the mobility binding carries OAuth2 client credentials scoped to the municipality and an entitlement condition for the live tier.

About the European Data Space for Smart Communities

The European Data Space for Smart Communities (DS4SSCC-DEP) initiative is a pivotal deployment following the preparatory action for a Data Space for Sustainable and Smart Cities and Communities (DS4SSCC). Emphasising sustainability aspects and diversity in communities, DS4SSCC developed a multi-stakeholder data governance scheme, created a blueprint for the European DS4SSCC, delivered priority datasets, and developed a roadmap towards a mature DS4SSCC.

DS4SSCC-DEP's vision revolves around creating a well-governed federated territorial, place-based data space for smart communities available for developers and infrastructure providers, aligning with the prospects outlined in Europe's Digital Decade objectives. This approach distinguishes itself from sectoral data spaces by encompassing diverse domains, and underpinning governance across all levels of society. The key objectives of the project are:

- **Objective 1: Establishing a Federated and Innovative Data Space**. This phase focuses on creating a large-scale data space controlled by public data holders, ensuring alignment with the Smart Middleware Platform and broader data space ecosystems. It also aims to offer middleware service solutions facilitating data sharing and management while refining the blueprint based on gained experience.
- **Objective 2: Pilots and Refinement of the Data Space Blueprint**. Through 10-12 cross-sectoral data pilots in various EU communities, this objective aims to validate and refine the data space blueprint. Using common high-value data sets, it seeks to create added value by combining data from various domains, contributing to the sustainability plan, and minimising infrastructure investment.
- **Objective 3: Fostering Innovation and Ecosystem Alignment**. Engaging stakeholders, compliance with sector-specific legislation, establishing links with Horizon Europe missions, creating services available via trusted application catalogues, and enhancing ethical considerations and AI-enabled local solutions constitute the focus of this objective.

DS4SSCC-DEP, building upon established European networks, aims to significantly contribute to the sustainability goals of European citizens. It emphasises co-creation, standardisation, business models, strategies for data spaces, and envisions a federated platform.

DS4SSCC-DEP consortium includes the following partners: