SIEMENS

SIMATIC IT Unilab 6.7

Concepts Guide - part 2

Concepts Guide

Edition 09/2014

A5E00428243-09

Guidelines

This manual contains notices intended to protect the products and connected equipment against damage. These notices are graded according to severity by the following texts:

Caution

Indicates that if the proper precautions are not taken, this can result into property damage.

Notice

Draws your attention to particularly important information on handling the product, the product itself or to a particular part of the documentation.

Trademarks

Disclaimer of Liability

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E00428243-09
09/2014

Preface

Where is this manual valid?

This manual is valid for release 6.7 of SIMATIC IT Unilab.

Basic knowledge required

This guide is intended for SIMATIC IT Unilab users who are responsible for system configuration, such as application managers and system integrators (consultants).

To be able to understand the concepts and examples discussed in this guide, the reader should at least have taken the SIMATIC IT Unilab Basic Training.

Purpose

This Concepts Guide explains all concepts of SIMATIC IT Unilab and includes several scenarios that illustrate how these concepts can be used in specific situations.

Related documentation

The **Unilab Concepts Guide part 1** and **3** contain information related to the content of this Concepts Guide.

These documents are all available online from the SIMATIC IT Unilab Documentation Library**.**

Conventions

The table below describes the specific typographic conventions that are used throughout this manual:

Symbol/Convention	Indicates...
E.g.	Where examples are given.
Text in bold	The names of menus, commands, dialog boxes and toolbar buttons and, in general, all strings (e.g. **File** menu; **Save** command).
KEY1+KEY2	Shortcut keys, which permit rapid access to commands (e.g. CTRL+C).
UPPERCASE	The names of keyboard keys (e.g. RETURN key).
Italics	Noun with special importance or significance for which emphasis is needed. The names of parameters that must be replaced with a specific name or value.
>	A succession of commands in which the command preceding the symbol must be selected before the

Symbol/Convention	Indicates...
	command following it.
`Code example`	Code example.

SIMATIC IT Documentation Library

The SIMATIC IT Unilab Documentation Library provides you with a comprehensive and user-friendly interface to access the overall product documentation where manuals and helps online can be browsed by functionality or by component.

Readme

The installation includes a readme file, which contains information on upgrade procedures and compatibility with previous releases. This file is supplied both in standard text (**Readme.wri**) and in Acrobat PDF (**Readme.pdf**) format.

This file is available in folder \ReleaseNotes of the setup DVD and is available from the SIMATIC IT Unilab Documentation Library.

Acronyms and abbreviations

The table below lists the acronyms and abbreviations that are used throughout this manual:

Acronyms / Abbreviation	Meaning
ad	Address
API	Application Program Interface
ar	Access rights
au	Attribute
ca	Intervention
cf	Custom function
cn	Condition
cs	Condition set
db	Database
DBA	Database Administrator
dd	Data domain
DLL	Dynamic Link Library
eq	Equipment
ev	Event
fa	Functional access
FIFO	First In First Out
freq	Frequency
gk	Group key
GUI	Graphical User Interface
hs	History
ic	Info card

Acronyms / Abbreviation	Meaning
id	Identification
ie	Info field (Configuration)
ii	Info field (Operational)
ip	Info profile
lc	Life cycle (or life cycle model)
LIMS	Laboratory Information Management System
ly	Layout
lo	Storage location
me	Method (Operational)
MES	Manufacturing Execution System
MRP	Materials Requirement Planning
mt	Method (Configuration)
pa	Parameter (Operational)
pg	Parameter group
pp	Parameter profile
pr	Parameter Definition
pref	Preference
pt	protocol
rd	Raw data
RDBMS	Relational Database Management System
rq	Request
rt	Request type
sc	Sample code
sd	study
seq	Sequence (number)
SOP	Standard Operating Procedure
spec	Specification
ss	Status
st	Sample type
tk	Task
tp	Time point
uc	Unique code mask
up	User profile
us	User
ws	Worksheet
Wt	Worksheet type

SIMATIC IT Training Center

Siemens IA AS MES offers a number of training courses to familiarize you with the SIMATIC IT product suite. To successfully achieve this goal, training consists of lessons in both theory and practice.

Courses are held year-round, according to a program that is published well in advance of the first scheduled session.

The material on the basis of which our courses are conducted reflects the result of years of experience in process, LIMS, quality control and production management.

All courses are held by expert personnel that are aware of the developments and innovations in the Siemens IA AS MES product suite.

Courses are held in English at the Siemens IA AS MES Training Centers.

Upon request, training courses can also be organized on the customer's premises.

For more information on the training course calendar, please visit our technical web site (http://www.siemens.com/simatic-it/training).

SIMATIC IT Service & Support

A comprehensive Software Maintenance program is available with SIMATIC IT products. Software Maintenance includes the following services:

- **Software Update Service** (SUS): automatic distribution of upgrades and service packs
- **Technical Support Service** (TSS): support on technical problems with SIMATIC IT software (standard support and other optional services)
- **Online Support**: a technical web site, providing information such as Frequently Asked Questions and technical documentation on SIMATIC IT products

Software Update Service (SUS)

This service provides automatic shipment of new versions and service packs when released. When a new version / service pack is available for shipping, it is typically shipped within one month.

One copy of the installation DVD is shipped for each Server covered by Software Maintenance.

Hot fixes (officially tested and released) are not shipped and can only be downloaded from the Technical Support Service web site.

Technical Support Service (TSS)

Siemens provides a dedicated technical support team for SIMATIC IT products.

The following options are available:

Bronze support: 9 hours/day, 5 days/week

Silver support: 24 hours/day, 5 days/week

Gold support: 24 hours/day, 7 days/week

The principal language of the SIMATIC IT hotline is English.

SIMATIC IT partners and customers covered by the Software Maintenance program are entitled to direct access to the TSS.

Access to TSS

To be able to access TSS, the customer needs to register as a user on the Technical Support web site: http://www.siemens.com/mes-simaticit/ and follow the **Technical Support Service** link.

The registration form must be completed with:

- Personal data
- The required company and plant information
- The Contract Number provided by Siemens Back Office when the contract is agreed.

Online Support

A customer who is a registered TSS user, can access the Technical Support web site (http://www.siemens.com/mes-simatic-it/tss), which contains technical information such as:

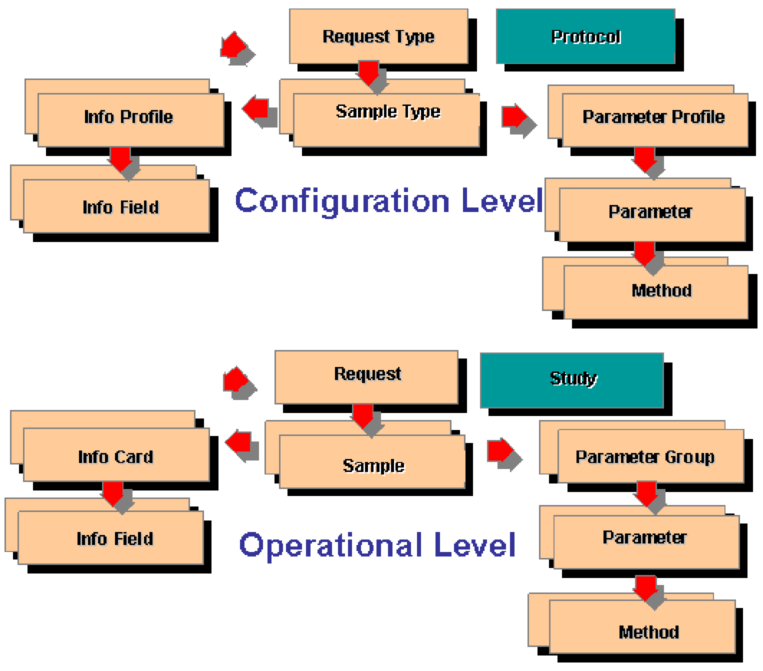
- Service conditions (Phone numbers, Working hours, Reaction times,...)
- SIMATIC IT knowledge base: a technical support database that includes practical service solutions from Technical Support or the SIMATIC IT community
- SIMATIC IT software (e.g. hot fixes, software examples) and release notes that can be downloaded
- SIMATIC IT cross-industry libraries that can be downloaded (limited access to SIMATIC IT certified partners)
- SIMATIC IT product documentation that can be downloaded
- Frequently Asked Questions and useful tips.

Table of Contents

1 Unilab Data Model

The following picture shows the data model of SIMATIC IT Unilab. Each item of this data model is an object.

The illustration below shows the Unilab data model.



Each of these objects is further detailed in the next sections.

Main objects of Unilab

The objects included in the data model are divided in two main categories:

- **Operational** objects, used when performing daily operations in the laboratory
- The properties and the behaviour of these objects are defined at the **configuration** level

For each object at the operational level, a corresponding configuration object exists. Each configuration object is a separate, independent entity. The properties can be set for each object individually, without interference from other configuration objects.

Main object pairs

The main object pairs of Unilab are:

- Sample and Sample Type
- Request and Request Type
- Info Cards and Info Fields
- Parameter Groups and Parameters
- Methods
- Studies and protocols

Sample and Sample Type

The central object in the Unilab data model is the sample. A sample can be defined in many different ways. Often, it simply represents a small part of material on which several analyses have to be performed and for which analysis results and/or other information must be kept.

For each sample, administrative details must be stored. This kind of information is maintained in the info card(s) of a sample. On each sample, a set of analyses is performed. These analyses, called parameters, are grouped into parameter groups.

A sample type represents a number of samples that have common characteristics. It is the template definition for a specific type of sample, used to facilitate the quick creation of new samples.

Request and Request Type

Samples are often processed collectively. Grouped processing requires its own work flow and information management. A link, which is quickly and easily accessible, should be maintained between samples of the same group. Examples of grouped processing are:

- A batch of samples arriving in the laboratory
- A complex sample on which several sub samples need to be taken
- An analysis request for which several samples need to be collected and analyzed.

A request corresponds to a group of samples that should be processed together. Administrative details with respect to the request are kept in info cards. The request has its own life cycle. Each of the individual samples included in the request can have their own info card.

Typical areas utilizing requests are:

- Analysis requests in service laboratories: the number and type of samples taken and the analyses to be performed on the sample depend on the analysis request by the customer
- Batch management: a production batch can only be released if all the samples taken from it lie within the specifications
- Complaint handling: when a complaint about a product is registered, the specific details of that complaint need to be processed. Often, several samples are taken from the product or even the production line to find out what is wrong
- R&D projects.

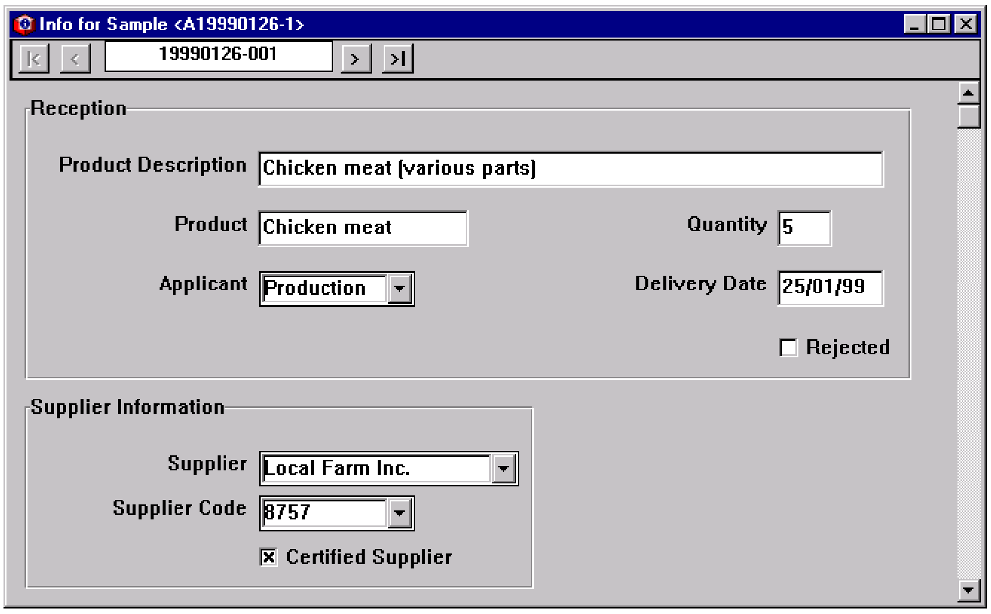
Info Cards and Info Fields

An info card contains all details that are specific to a sample or request. Info card details can be filled out and consulted at any moment. Info cards are mainly used to enter administrative details for samples or requests.

Each info card contains a number of info fields. Info fields are the building blocks of info cards. They contain general sample information, e.g. sampling location, batch number, supplier, sampling date, delivery date, etc.

Several info cards can be assigned to a sample. For instance, the info card for sample reception, the info card containing supplier information and the info card for sample approval, where conclusions can be entered by the lab manager after the requested analyses have been performed.

The picture below shows the info cards **Reception** and **Supplier Information** for sample 19990126-001.



Info for Sample <A19990126-1>

19990126-001

Reception

Product Description	Chicken meat (various parts)		
Product	Chicken meat	Quantity	5
Applicant	Production	Delivery Date	25/01/99
			☐ Rejected

Supplier Information

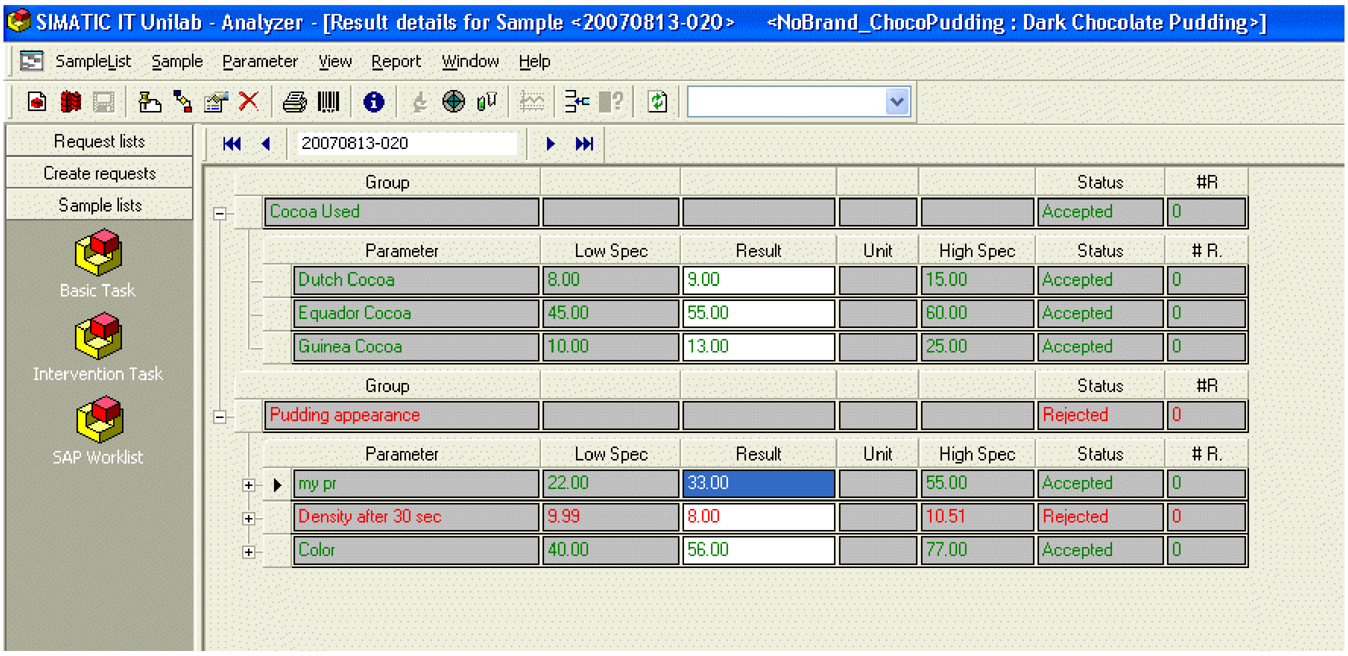
Supplier	Local Farm Inc.
Supplier Code	8757
	☒ Certified Supplier

Parameter Groups and Parameters

A specific set of analyses must be performed on each sample. The sample property that is determined through analysis is called a parameter. Each analysis provides a parameter result. The parameters can be physical, chemical, microbiological or other properties of the sample.

Parameters that logically belong together, such as a set of analyses performed in the same section of the laboratory, or all the analyses of the same type (chemical analyses, microbiological analyses) performed on a sample, are grouped into parameter groups.

The picture below shows the **Parameter result** window displaying the different parameter groups and parameters for the sample **20070813_020**. The parameter groups are **Cocoa Used** and **Pudding appearance**.



Group					Status	#R
Cocoa Used					Accepted	0

Parameter	Low Spec	Result	Unit	High Spec	Status	# R.
Dutch Cocoa	8.00	9.00		15.00	Accepted	0
Equador Cocoa	45.00	55.00		60.00	Accepted	0
Guinea Cocoa	10.00	13.00		25.00	Accepted	0

Group					Status	#R
Pudding appearance					Rejected	0

Parameter	Low Spec	Result	Unit	High Spec	Status	# R.
my pr	22.00	33.00		55.00	Accepted	0
Density after 30 sec	9.99	8.00		10.51	Rejected	0
Color	40.00	56.00		77.00	Accepted	0

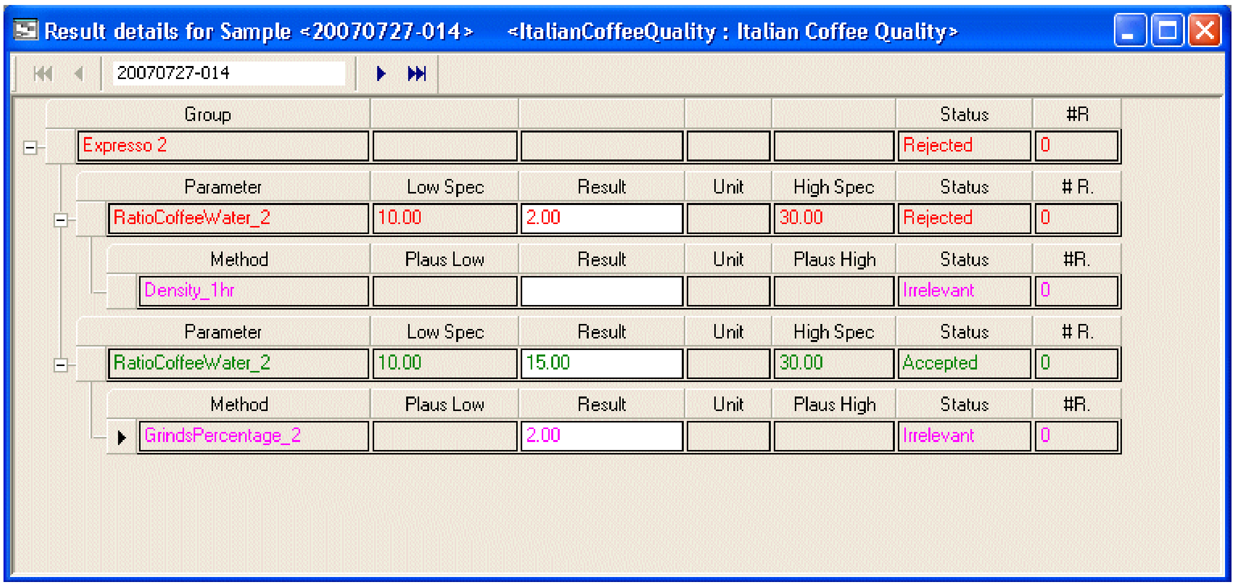
For each individual parameter in a parameter group, a number of specifications can be defined. These specifications may have different values for the same parameter, depending on the parameter group to which that parameter belongs.

Each parameter result is validated against the specifications. An alarm can be triggered when the parameter result falls outside the defined specification range. In the example above, the parameter result for parameter **Density after 30 sec** is not acceptable (i.e. rejected), as it falls below the **Low Spec.** value.

Methods

A method defines how a parameter result is determined. In certified labs, procedures for the execution of methods are written in ISO 9000 or GLP documents (Standard Operating Procedures or SOPs).

Often, several methods or techniques can be used to analyze a sample property (parameter). In the example below, parameter **RatioCoffeeWater_2** is measured using two different methods: **Density_1hr** and **GrindsPercentage_2**. In some laboratories, one method may be defined as the standard one, another method may have to be used to check the results of the first method, and a third method may have to be used if a higher degree of accuracy is required.


Result details for Sample <20070727-014> <ItalianCoffeeQuality : Italian Coffee Quality>

20070727-014

Group					Status	#R
Expresso 2					Rejected	0

Parameter	Low Spec	Result	Unit	High Spec	Status	# R.
RatioCoffeeWater_2	10.00	2.00		30.00	Rejected	0

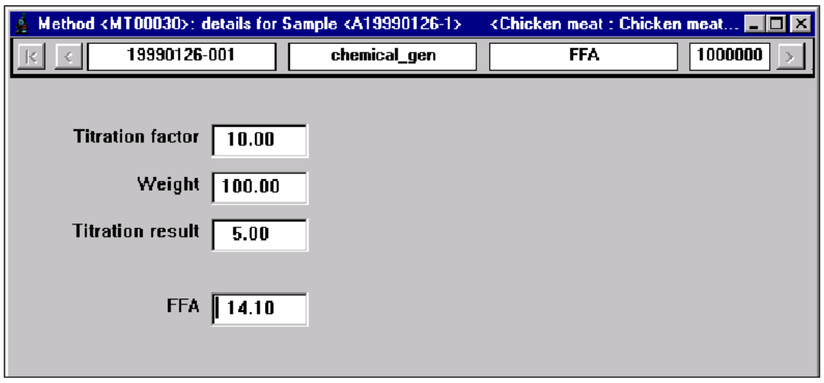
Method	Plaus Low	Result	Unit	Plaus High	Status	#R.
Density_1hr					Irrelevant	0

Parameter	Low Spec	Result	Unit	High Spec	Status	# R.
RatioCoffeeWater_2	10.00	15.00		30.00	Accepted	0

Method	Plaus Low	Result	Unit	Plaus High	Status	#R.
GrindsPercentage_2		2.00			Irrelevant	0

The basic structure for method sheets is a free-form window, which needs to be configured to contain the data that is relevant for each specific method.

The picture below shows the method sheet for the titration method (MT00030) used to determine the Free Fatty Acid content (FFA) of a sample.


Method <MT00030>: details for Sample <A19990126-1> <Chicken meat : Chicken meat...

19990126-001 chemical_gen FFA 1000000

Titration factor	10.00
Weight	100.00
Titration result	5.00
FFA	14.10

Studies and protocols

As from SIMATIC IT Unilab 6.1, dedicated support for **stability** testing is provided. Stability testing is a routine procedure performed on drug products, active ingredients and excipients in order to understand the long-term effects of the environment on the drugs. It is used at various stages of product development. In the early stages, accelerated stability testing (high temperatures and relative humidity) is used to evaluate the stability of the drug and to determine what types of degradation products may be found after long-term storage. Testing under normal conditions (those recommended for long-term shelf storage) can be used to determine a product’s shelf life and expiration date. In these types of studies, the product is analyzed at predetermined intervals for various parameters.

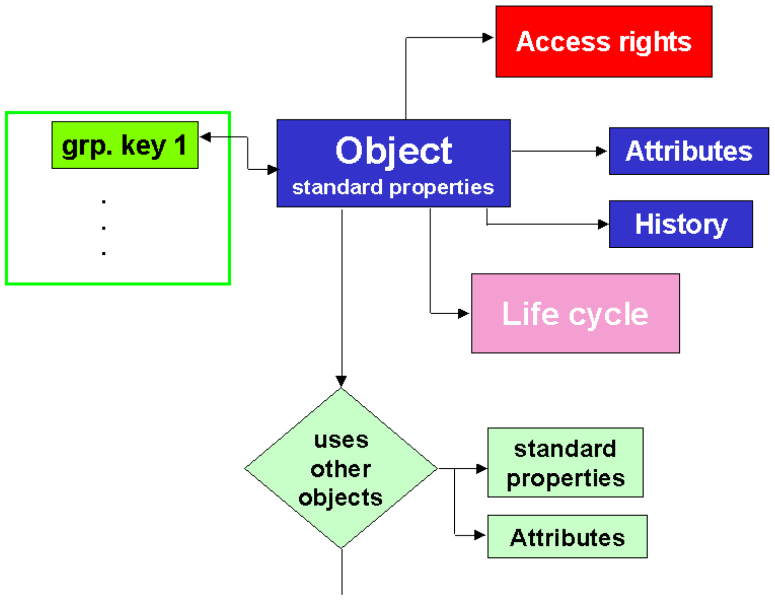
A **protocol** is the main object for the configuration of a stability test.

A **study** is the actual execution of a stability test.

1.1 Generic Object Design

All Unilab objects have the same generic structure, resulting in a system that is easy to modify and extend.

The picture below shows the generic object structure in Unilab.



Most of the common characteristics of an object are defined in the object property sheet. Each object has a number of standard properties that apply to every object of that type. These properties are set by means of standard attributes. The standard attributes can be extended with:

- User-defined attributes
- A life cycle
- Access rights
- An audit trail (history)
- Group keys (when applicable).

1.2 Concept of Used Objects

A number of used objects can be assigned to each main Unilab object. For instance, parameter profiles and info profiles can be assigned to a sample type. Parameter definitions can be assigned to a parameter profile. Methods can be assigned to a parameter definition.

The list of used objects is part of the properties of a main object.

Important

It is possible to define some new properties on the link between a main object and a used object. Some new properties may overrule the underlying properties of the used object itself.

On the link with used objects, some additional standard properties can be set. The standard link properties can also be extended with user-definable properties.

Used object relations

All used object relations within the configuration model are one-to-many relations. Multiple parameter profiles can be assigned to one sample type. Multiple parameter definitions can be assigned to one parameter profile.

In accordance with a general rule for one-to-many relations, the number of used objects is not limited to a specific minimum or maximum. In other words, many should always be understood as zero, one or more. This means that it is perfectly acceptable not to assign info profiles to a sample type if no administrative details are required for samples of that sample type.

Modular behaviour of Unilab configuration

The Unilab configuration behaves in a modular way. Once configured, an object can be used in various used object lists. For instance, once a parameter profile has been defined, it can be assigned to different sample types; different sample types may refer to the same parameter profile. A parameter definition can be assigned to different parameter profiles (with different specifications).

The concept of modularity is valid at every level, even at the method level. Some methods, such as a chromatography, can be used to determine different parameters. The method sheet needs to be defined only once; it can then be assigned to different parameter definitions.

Properties of used objects

Modular systems may lead to decreased flexibility. Although a parameter definition can be used in different parameter profiles, some of its properties may be different according to the profile it is assigned to. Parameter specifications may differ according to the parameter profile to which the parameter is assigned.

To assure both modularity and flexibility, Unilab offers the possibility to define properties on the link main object–used object. Typical examples of such properties are:

- The assignment frequency, i.e. the frequency with which the used object is assigned to the main object
- The specifications of parameters within a parameter profile.

Some properties defined on the link main object–used object overrule the corresponding properties of the used object. For example, a delay can be specified in the property sheet of a parameter. This delay is added to the delay specified at parameter profile level. The delay of the parameter can therefore be reset on the link parameter profile–parameter.

1.3 Configuration versus Operational Level

Despite the similarities in the model at both levels, there are some differences in referential integrity between configuration and operational objects. These functional differences result in technical differences on the database side.

Object definition versus operational object

First of all, functional differences exist between configuration objects and objects at the operational level. Configuration objects are actually object definitions: they contain the definition of an object at the operational level. Consequently, the objects act as separate entities. The properties can be set for each object individually; there is no interference with other configuration objects. The operational model on the other hand provides a framework for the entry of lab data. With the exception of samples and requests, objects are only referred to within their referential context. Parameters, for instance, can only be consulted in connection with the sample they are assigned to.

These functional differences cause the same action to behave differently at the configuration and operational level. When a parameter definition is deleted, all instances in parameter profiles in which that parameter is referenced are deleted as well. On the other hand, the deletion of a parameter at the operational level is only valid for the selected sample.

For information on the relationship between objects and data access rights, see par. “Accessing Configuration and Operational Objects” in the **SIMATIC IT Unilab - Concepts Guide Part 3**.

Active Allow_modify flags

The functional differences between configuration and operational level are also reflected in the interpretation of the **Active** and **Allow_modify** flags.

Note

For more info on how to interpret the **Active** and **Allow_modify** flags, refer to the chapter on life cycles.

Sequences and nodes

Special structures are implemented to uniquely identify the objects used at both levels. However, these structures are not the same at configuration and operational level. At the configuration level, sequences are used. At the operational level, used objects are uniquely identified by means of node numbers.

Sequence Numbers at Configuration Level

Sequence numbers are just numbers (generated in a PL/SQL loop) to uniquely identify each row in a table. They are represented by a numeric value with a maximum of three characters. Sequence numbers have two primary functions:

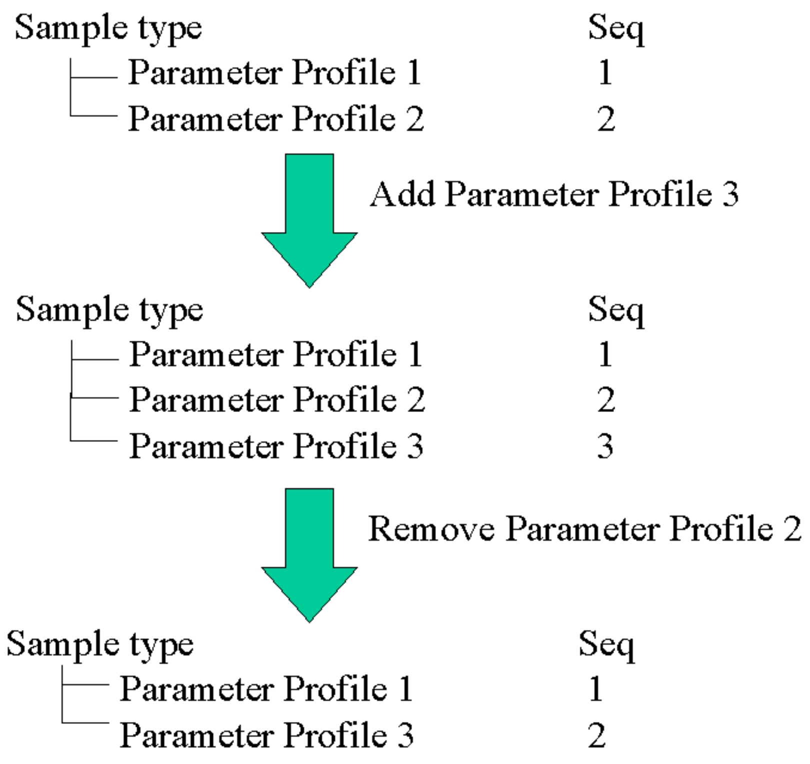
1. They represent the order in which the used objects will be displayed on the screen
2. They allow for multiple assignment of the same object.

Sequence numbers are not static, they can dynamically change through the existence of the object. When a used object is added/removed, the sequence numbers of the other used objects are modified accordingly. On assignment/removal of a used object, the DB API deletes the existing list and creates a totally new list with appropriate sequence numbers. Therefore, sequence numbers are not used for explicit identification of a used object.

Example

When added to a sample type, parameter profile 3 receives the sequence number that corresponds with its place in the list. When another parameter profile is removed from the list, the list of assigned objects is recreated and all the sequence numbers of the other parameter profiles are adjusted accordingly.

The picture below shows the re-sequencing during the assignment/removal of a used object.



Node numbers at operational level

At the operational level, node numbers are used to uniquely identify each entry. Node numbers exists at each level: parameter group (pgnode), parameter (panode), method (menode), info card (icnode), info field (iinode) and raw data (rdnode).

Node numbers have a maximum length of 9 characters, with a default increment of 1,000,000. Nodes add up downwards. Each node number refers to only one level (e.g. pgnode defines the relation between a sample code and a parameter group). This supports the multi-level hierarchy (sample code – parameter group – parameter – method).

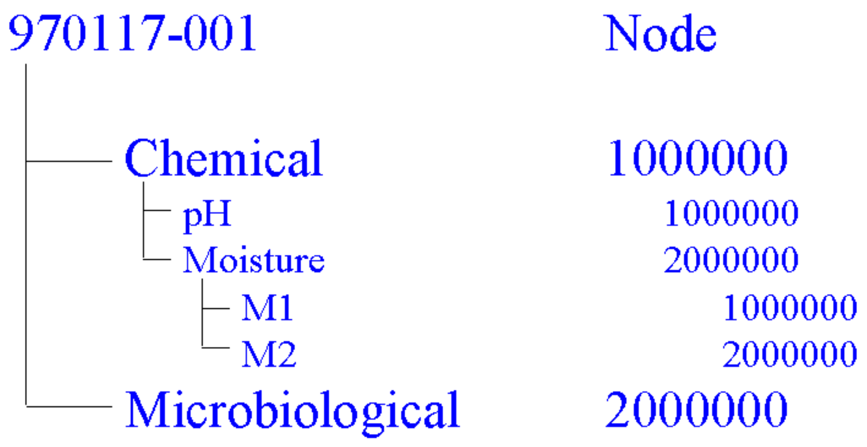
Important

An exception to the use of node numbers at the operational level is found on the relation between samples and requests. This combination is made unique by means of sequences, not by means of nodes. In this particular case, the sequences only represent the display order of the samples in the request. It is possible to add/remove samples to/from a request. The sequence number of the other samples in the request will then be modified accordingly.

The keys to each object are its ID(s) and the nodes on the different hierarchical levels.

- e.g. sc + pg + pgnode + pa + panode
- e.g. sc + ic + icnode + ii + iinode.

The picture below shows an example on the use of node numbers at operational level.



Node numbers are also used to determine the actual order of the entries. This is because the order of the entities cannot be based on the alphabetical order of the names of the entities. Moreover, the names of the used objects at the operational level are not necessarily unique. The same object (for example, a parameter) can be assigned more than once to the same parameter group. Two measurements of the same parameter are then performed. The two measurements usually produce a different result. Node numbers remain the same during the entire life of the object. In addition,

- Up to 20 levels can be inserted before a specific node number
- A node number can be inserted between two objects that are assigned already
- A new node number can be appended at the end of an existing list.

2 Object Properties

This chapter describes a number of generic properties. These are properties that apply to all Unilab objects (both operational and configuration objects).

Identification of configuration objects

At every level, configuration objects are uniquely identified by a name. Because the name serves as unique identification for the object, it cannot be modified once it has been saved to the database.

In addition to the name, two descriptions are provided for each object. Many laboratories apply a description that is internally used within the company and a description for external use. Therefore, two descriptions can be defined at the configuration level. The second description is never used within Unilab itself, it is only used for reporting purposes.

Identification of operational objects

The first description defined for a configuration object is always inherited by the corresponding operational object. This description cannot be used to uniquely identify the operational object because for each sample type (request type), multiple samples (requests) are logged on. Samples (requests) are uniquely identified by their sample code (request code).

All other objects at the operational level are uniquely identified by means of the objects at higher hierarchical levels and the node numbers at every level.

Example: A parameter at the operational level is uniquely identified by the following combination: sample code + parameter group + pgnode + parameter + panode.

Use as template

Every configuration object can be used as a template for the creation of more objects of the same object type. When creating new objects of a certain object type, the user has the choice between creation from scratch (all properties still have to be defined) or creation based on an existing template (some properties are copied from the template).

All objects (active and inactive) can be used as a template.

Example: Sometimes it is necessary to create multiple sample types with similar properties. To speed up the configuration process, Unilab offers the possibility to configure a sample type as a template. Creating new sample types can then be done using the existing template, which helps reduce configuration errors.

In case a template is used for the creation of new objects, the following data is copied from the template (if relevant for the object type):

- All template standard properties are copied, but the template flag itself is switched off and the status is also cleared (because the life cycle will again start from the beginning for the newly created sample type)
- User-defined attributes are copied
- Group keys are copied (only relevant for sample and request types)
- The list of used objects and all attributes of these assigned objects are also copied (e.g. the list of info profiles and parameter profiles for a sample type).

The **What** part of the first entry in the history logging of an object, which has been created from an existing template will always be of the form:

<Object type> has been created from <template id>

Thus it is always possible to track how the object was created.

Important

Templates are used to copy the properties of one configured object to another. Templates do not provide links fur future configuration maintenance. For example, once a sample type has been created based on a template, further modifications to the template will not be reflected in the sample type.

User-definable attributes

Unilab allows considerable control over the setup of the laboratory's information system. Most object characteristics can be defined in the object’s property sheet. Each object has a number of standard properties that apply to every object of that type. These properties are set by means of standard attributes. Examples of standard attributes are the sample code mask for a sample type or the plausibility range associated with a method.

Nevertheless, certain laboratory requirements may need even more control and fine-tuning of the various object definitions in the information system. Subsequently, the list of standard object attributes can be extended with a list of user-definable attributes. These attributes can represent key information for reporting purposes (e.g. the ISO method name, the sample’s commercial name or the object name translation in other languages). Attributes can also represent a **Microsoft Word** document, an **Excel** file or any other file that can be made accessible from within Unilab. For example, this approach can be used to make a chromatogram, related to a specific method sheet, easily accessible from within the method sheet itself.

User-defined attributes can be assigned:

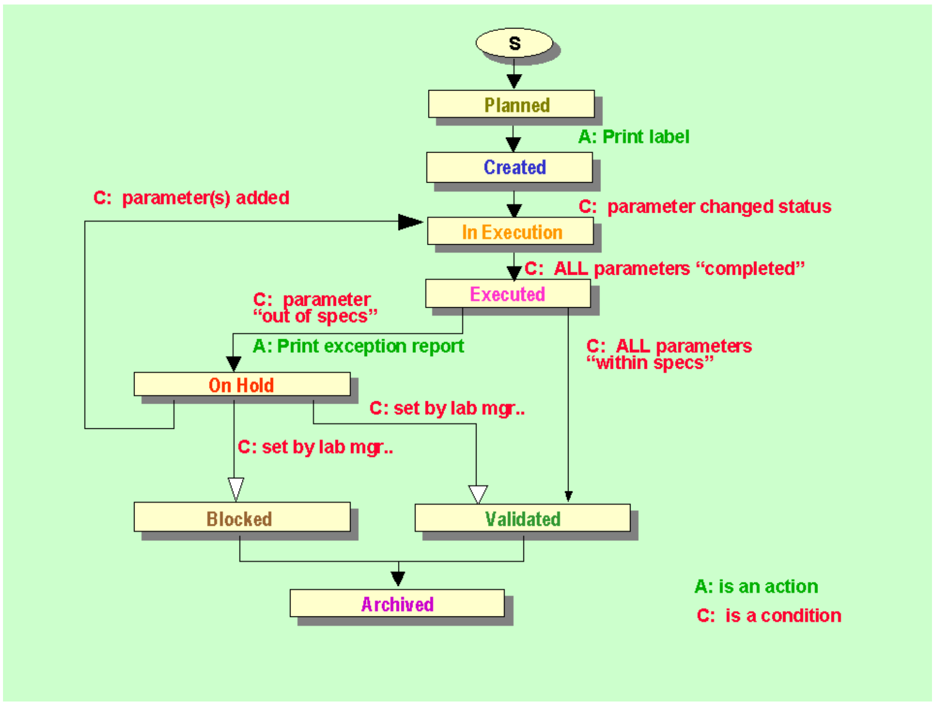
- To each configuration object (e.g. to a sample type)
- To each relation between objects (e.g. to the sample type – parameter profile link).

The corresponding operational object can inherit attributes set on the configuration level. For instance, the commercial name of a marketed product can be set as a value for a sample type attribute. This attribute and its value can be inherited by the samples of the sample type.

Life cycle

All Unilab objects have their own life cycle. A life cycle model is a complete "state transition diagram" specifying the conditions that trigger an object to change from one state to another state and the actions that are executed when the state transition occurs.

The picture below shows an example life cycle model for a sample.



The object life cycle object consists of a number of object states and state transitions.

The state of an object determines whether an object:

- Is active or not.
- Is modifiable or not.
- In what color the object (text) will be shown when "listed".

The transition from one state to another can be triggered manually, or can be triggered automatically as soon as certain conditions are fulfilled.

For each manual transition within the life cycle model, only a limited (specific) number of people have the authorization to initiate the transition. An automatic transition is triggered as soon as a specific condition is met.

The transition itself can trigger some actions, such as the generation of an exception report or the creation of another object.

At the operational level, each object has its own life cycle. Although the life cycles of different objects (sample code, parameter group, parameter and method) are not the same, they closely interact. After all parameters in a parameter group receive the status **Validated**, the parameter group itself becomes validated. When a sample is cancelled, all assigned parameter groups, parameters and methods also go to the **Cancelled status.**

Important

The life cycle work flow control entities are not actually part of each object. These entities are treated as objects themselves. Actually, each object contains a pointer to a specific life cycle.

Access rights

For each configuration object, the access rights for each of the data domains (maximum 128 datadomains) can be set to:

- **READ** (users belonging to current data domain can only view the object)
- **WRITE** (users belonging to current data domain have read-write access)
- **NONE** (users have no access to the object; the object will never be displayed in lists viewed by users belonging to current data domain).

The initial access rights for newly created operational objects are not automatically inherited from the corresponding configuration objects. The strategy used for initialization of access rights at the operational level for users of a particular data domain is defined during database installation.

Audit trail

An audit trail is kept and maintained for each object. Audit trails keep track of any entries or changes to the objects. The audit trail logging keeps a full audit trail for the object:

- What happened with the object?
- When did the modification occur?
- Who was responsible for the modification?
- Why was the modification done?

Example: When updating a property of the object, the database will log the following:

```
“<object type> "<object id>" is updated: property <property
description> changed value from "<old value>" to "<new
value>".
```

Note that the <object type> is always a "readable description” The “who” description always contains the name of the user who performed the transaction. This can also be an application. In the case, the “who” description holds a “readable” description of the job/application or additional details about the task that this application was performing.

The timestamp in the audit trail and any automatically registered date/time information is always using the unique Oracle date/time of the server!

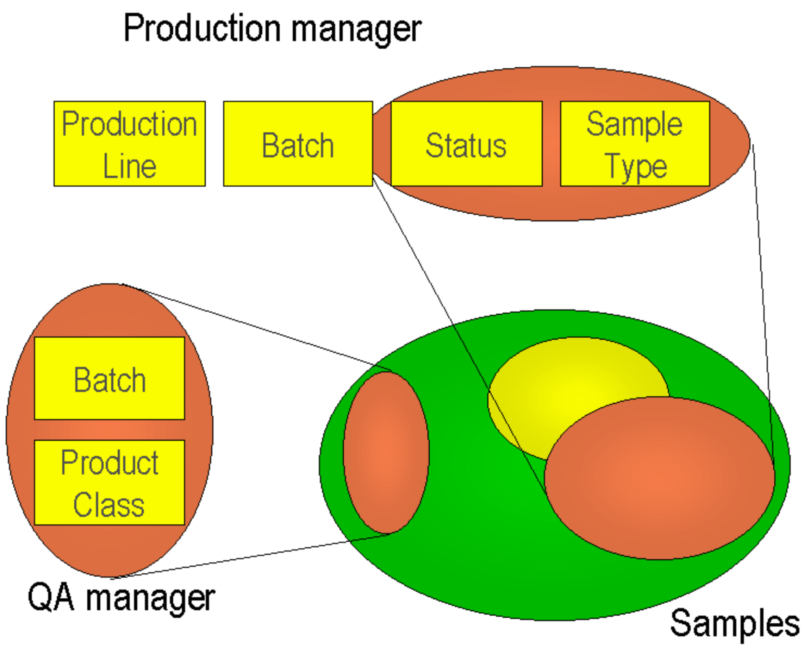
If no audit trail logging is required, this option can be switched off. However, some items will always be logged:

- The fact that the audit trail logging was switched off
- Object-specific comments that were explicitly entered (through the Add Comment functionality)
- Electronic signatures.

Selecting objects in Unilab

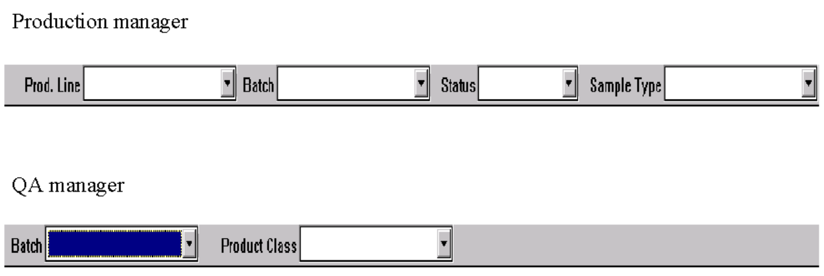
Different Unilab users have their own requirements to access laboratory information. For example, a production manager might be interested in samples from a certain production line, samples of a certain type, samples belonging to a certain batch (**Batch**) or samples in a certain state (e.g. **in execution** or **on hold**). On the other hand, a QA manager might be interested in samples of a certain product class (**raw materials**, **finished products**), or samples of a certain batch (**batch**).

The figure below illustrates the group-key concept.



Selection of the specific data that a user is interested in can be done by means of group keys. Group keys are selection keys that enable a Unilab user to filter the data retrieved from the database in a way that combines flexibility and performance. The group-key bar behaves like a second customizable toolbar that contains drop down list boxes instead of buttons. Each user gets his own keys or search paths to navigate through a list of samples or objects.

The figure below provides examples of the group-key bar for different users.



Group keys can be defined for requests and samples. At the configuration level, group keys only exist for sample types,request types and protocols. For methods at the operational level, group keys are used for worklist assignment rules.

The use of group keys and assignment rules are different for each object type. More details concerning the application of group keys for the different object types are provided in corresponding chapters.

In specific contexts, there also exists the possibility of using the group-key pane, which employs Boolean and comparison operators, to narrow down the list of objects that you want to consult. For details on both the group-key pane and the group-key bar, see par. “Use of Group Keys in the Task Bar” in the **SIMATIC IT Unilab - Concepts Guide Part 3**.

3 Samples and Sample Types

3.1 Introduction

3.1.1 Samples

A sample is a small part or portion of a material or a product intended to be a representative of the whole. By analyzing the properties of the sample, laboratory people aim to establish a clear view on the properties of the material or the product that was sampled.

Each sample is uniquely identified by means of a unique sample code. Administrative information on the sample is kept in info cards. The analyses performed on the samples, the parameters, are grouped into parameter groups, which are assigned to the sample.

3.1.2 Sample Types

To minimize data entry during the creation process of new samples, the properties of samples with common characteristics are defined in a sample type. A sample type thus represents a number of samples that have some common characteristics.

The criteria to identify or define different sample types vary from one laboratory to another. In some cases, the sample types correspond to physical product types. For example, when a veterinarian takes samples from different animal types, like chickens, rabbits, dogs, cats, pigs and cows, these animal types are most likely candidates to be defined as sample types. Each sample taken from another cat is unique, but has a lot of characteristics in common with all other cats. It would not make sense to test a cat for diseases that can only occur in a cow. The (official) identification systems for cows, pigs and dogs are also different, so each info card can be configured accordingly (this expedites data entry and helps avoid errors).

In other cases, the sample type holds a reference to the place where the samples were taken or the conditions under which the samples where obtained. In a waste water installation for instance, all samples coming into the laboratory are water samples. The properties of the sample however differ depending on the place in the installation were they were taken. Therefore, it does not make any sense to define a sample type **Water**. Instead, the sample types may refer to the pump or valve where the corresponding sample was obtained.

3.1.3 Sample Group Keys

Further grouping of the samples can be done based on sample characteristics such as location, farmer, sampling date or priority. This grouping is implemented by means of group keys. Group keys also provide a fast and efficient way to retrieve a list of samples having common characteristics.

3.1.4 Analysis Requests

Often, samples need to be processed in a grouped way. Moreover, grouped processing requires a work flow and its own information. A fast and accessible link must be maintained between those samples. Examples of such grouped processing are a number of samples taken on the same production batch or a complex sample that is split into several sub-samples (or aliquots). Such grouped processing can be handled through requests.

3.2 Sample Properties

The majority of sample properties are copied from its sample type. However, some properties are managed at the sample level as well.

3.2.1 Identification of Samples

Each sample is uniquely identified by means of a sample code. The code generation algorithm for sample codes is defined by means of a unique code mask. The unique code mask that is used for samples of a certain sample type is included in the sample-type definition. For each sample type, a different code masks can be set. The sample code then depends on the sample type to which the sample belongs.

Important

For samples belonging to a request, yet another code mask can be applied. This mask, set at request-type level, always overrules the code mask set for the sample type itself.

The sample code is the unique sample identifier in the database. This implies that a sample code is always unique for one sample, no two samples can have the same sample code. Since the sample code is the primary key for the sample, it can never be modified.

3.2.2 Sample Description

Sample properties always include a reference to its sample type and description. Many laboratories apply both a company internal and an external description. Therefore, two descriptions can be defined at the sample-type level. The second description is not duplicated to the operational (sample) level, but is used for reporting purposes. For example, the second description can be displayed on certificates or analysis reports sent to external parties.

3.2.3 Sample Creation and Processing Details

Many sample property values are entered during sample creation. Some of these provide details on the creation process itself, such as creation date (the date the sample was logged into Unilab) and the ID of the user who created the sample. When a sample was created within a request, the request ID is supplied as well. Another field holds the sampling date. By default, the sampling date is set equal to the creation date. Often, samples are taken (long) before they are logged into Unilab. Therefore, the sampling date can always be modified.

Some sample properties are inherited from the sample type. This is the case for the sample shelf life limit, the priority, or the standard operating procedure that describes how the sample should be handled. Although these properties are inherited from the configuration level, there are ways of modifying them at the operational level.

Sample processing details are provided by its current status in its life cycle. The default life cycle model used for newly created samples is defined at sample-type level. Authorized users can always assign another life cycle model to a sample (although this is only done sporadically). The analyses start (entering the parameter results) and their eventual completion is also supplied automatically.

3.2.4 Optional Date and Time Info

Besides the sampling date and sample creation date, other dates might provide critical information on samples. Examples are the execution end date or the date on which the certificate was generated. For each sample, five additional dates can be retained. The labels of these optional date fields are configured system-wide using the system settings **LABEL_SC_DATE1** to **LABEL_SC_DATE5**.

The additional date info can be entered manually in the sample’s property sheet or the value can be set as a default value for an info field in one of the sample’s info cards. This default value links the info field to the optional date fields.

The value of the optional date info fields can also be populated by an action triggered via a state transition.

Example: The label for date 1 can be set to Certification date. In this field, the current date is supplied by an action on the life cycle state transition from Validated to Reported. The state transition itself is triggered by the generation of the analysis certificate for the involved sample.

3.2.5 Barcode Labels

After arriving in the laboratory, samples are often labeled for identification. Samples belonging to a request can even receive two labels: one to identify the request and one to identify the sample within the request. The label can be alphanumeric, a barcode or a combination of both. Most labels include barcodes.

The definition of a barcode label and the connection to the printer is handled through the definition of a Visual C++ custom function (stored on the client side). The label print function depends on the barcode reader, which defines the preamble and the applied barcode standard. The barcode label applied to samples of a specific sample type is defined in the sample-type properties by selecting the appropriate barcode custom function.

3.2.6 Modifying the Sample Properties

Sample properties can be modified at the operational level on the condition that the current sample status allows modification. There are four ways of doing so:

- In the **Property** tab of the sample itself
- Through info fields in info cards
- Through life cycle actions
- Through the sample list layout.

In the Property tab of the sample

Users having appropriate access rights can modify the sample properties in the property tab of the sample itself. However, some fields in this tab are protected and cannot be modified. For example, this is the case for the creation date or the ID of the sample creator.

Through info fields in info cards

Sample properties can also be modified through info fields on the info cards that are assigned to the sample. For each info field, a default value can be specified. This default value can be a standard sample property or the value of a user-defined attribute of the sample or the sample type. This default value actually links the info field to a specific standard property or attribute. Modifying the info-field value automatically modifies the standard property or the attribute value of the sample.

Important

Some standard properties of the sample are read-only and cannot be modified for example the sample code and the sample creation date.

This mechanism can also be applied to visualize certain sample properties to users without giving them access to the sample properties themselves (= selective access). By setting the protected field for the involved info fields, unauthorized modification of the properties can be avoided.

Through life cycle actions

Some sample properties change in conjunction with the life cycle or need to be modified based on certain state transitions. For example, this might be the case for the sample priority.

Through the sample list layout

The sample list layout can accommodate of sample properties. The sample properties are then displayed as columns in the layout and can be edited here as well (as long as the corresponding column is not protected).

3.2.7 Info cards and Parameter Groups

A sample has a list of parameter groups and info cards assigned to it. These objects are either assigned manually or automatically (based on the specified assignment frequency).

3.2.8 Sample Group Keys

A sample can have a list of group keys assigned to it. The group-key list can be established based on:

- Group-key inheritance from the sample type
- Sample group-key assignment rules.

Group keys inherited from sample type

Sample group keys and their values can be inherited from the corresponding sample type. This can be obtained by setting the Inherit flag on sample group-key level itself.

Group keys for which this flag is not set, can still be inherited by the samples of a specific sample type. This is done by setting the **Inherit group keys** flag on sample type level.

Tip

For more info on the inherit mechanism for group keys is referred to the chapter on Inherit Group Keys in **SIMATIC IT Unilab – Concepts Guide - Part 3**.

Group-key assignment rules

Upon sample creation, a number of group-key assignment rules are evaluated. When a rule applies, the corresponding group key is assigned to the sample with the appropriate value. Typical examples of group keys that are assigned on sample creation are the group keys **Sample type** and **Creation date**.

3.3 Sample-type Properties

A sample type is the template definition for specific types of samples used to speed up and simplify the creation of new samples. In addition, some settings are provided to control the properties of the sample type itself.

3.3.1 Properties for Samples

The sample-type property sheet includes a number of settings that define the properties and the behaviour of the samples at the operational level. Each of these properties is set manually at the configuration level.

These settings include:

- The default life cycle model for samples of this type
- The sample code mask used to generate the sample codes
- The standard operating procedure describing how to handle samples of this type
- The default priority for these samples
- The sample shelf life limit at operational level
- The label format used for the samples.

For reasons of security and the audit trail, the user ID may be confirmed upon sample creation. This feature can also be defined at the sample-type level.

3.3.2 Sample-type Group Keys

For sample types, a list of group keys and their values can be defined. These group keys serve multiple purposes.

First of all, group keys are used for grouping sample types based on their characteristics such as material type, supplier, or customer. This way, the group keys provide a fast and efficient way to retrieve a (specific) list of sample types having common characteristics.The sample-type group keys are also used during the new sample creation process. To make the identification of samples in the sample creation window easier, only the sample types are displayed that match the selection in the user-defined group keys.

Finally, the sample-type group keys can be inherited by samples belonging to the sample type. In this way, sample-type group keys can be used to restrict the sample list to a meaningful set on the operational level.

3.3.3 Sample Planning

In certain situations samples are to be taken at regular time intervals. The type and number of samples to be taken and their sampling frequency can be implemented in Unilab by means of sample planning. Sample planning includes two topics;

- Setting up the sample plan by defining the sample planning frequencies
- Executing the sample plan by running the sample planner.

Sample planning frequency

Samples requiring a certain frequency can be configured in the sample-type properties.

In most cases the time based frequencies will be applicable (schedule a sample of this sample type one per x days, y weeks or z months). Sometimes site-specific frequencies should be used, which can be implemented using custom frequencies.

It is also possible to specify the number of samples to plan. The application manager can edit the last scheduled date to synchronize the creation of samples of different sample types.

Running the Sample Planner

The **Sample Planner** must be run to generate the sample plan for a specific period. The **Sample Planner** is a Unilab background process that automatically schedules samples according to the defined sampling schemes.

Samples that are created by the **Sample Planner** always get the status **Planned** (@**P**).

3.3.4 Used Objects for Sample Type

List of used objects

Multiple info profiles and parameter profiles can be assigned to a sample type.

On the link “sample type–used object”, various properties can be defined, such as the assignment frequency. Some properties of the used object themselves can be overruled. For example, it is possible to make an info profile, that is by default not mandatory, mandatory for a specific sample type.

Assignment of used objects at operational level

During sample creation, used objects (info cards and parameter groups) will be automatically assigned according to the assignment frequencies defined at the sample-type level.

Circumstances may require manual assignment of used objects. The analyst can then make a selection from the list of used objects. For the parameter profile, this list might prove to be too restrictive. At the sample-type level, it can therefore be specified that the user is allowed to assign other parameter profiles to a sample of this sample type than those specified in the sample-type definition. The user then has the total list of active parameter profiles at his disposal for assignment.

3.4 Sample Creation

The logon of samples can be done manually or automatically. An analyst manually creates individual samples in the analyzer application.

Request creation can lead to batch creation of samples: the creation of a new request often includes the automatic creation of a number of samples.

The automatic sample logon can be event driven or time based. Event driven sample logon can be implemented through actions in the object’s life cycle. An example is the creation of new samples in case analysis results (parameter results) on other samples prove to be out of spec. Time based sample logon is executed based on an established sample plan.

3.4.1 Generation of the Sample Code

For each sample that is logged on, the system automatically generates a unique code, based on the unique code mask defined for the corresponding sample types.

Some of the sample properties are supplied at sample creation. This includes the creation date and the ID of the sample creator. Also the sampling date is supplied automatically. The default value of the sampling date is controlled through the **ScCreateSamplingDate** preference. In contrast to the creation date, the default sampling date can still be modified after the sample creation process has terminated.

3.4.2 Group-key Initialization

On creation of a sample, the appropriate group keys will be initialized for the sample. Some group keys are assigned based on established group-key assignment rules. The group keys and their values, specified at sample-type level, can also be inherited by the samples.

3.4.3 Attribute Assignment

Attributes defined at the sample-type level can be inherited by the corresponding samples. Inherited attributes can be set on two levels:

- On sample-type level (through the **Inherit attributes** flag)
- On attribute definition level (through the **Inherit Attribute** property).

The inherited attributes are assigned at sample creation.

3.4.4 Info Card and Test Plan Assignment

By default, the info cards and the test plan are assigned to the samples at sample creation. The test plan includes all the parameter groups, parameters and methods to be assigned to the sample, taking into account the assignment frequencies.

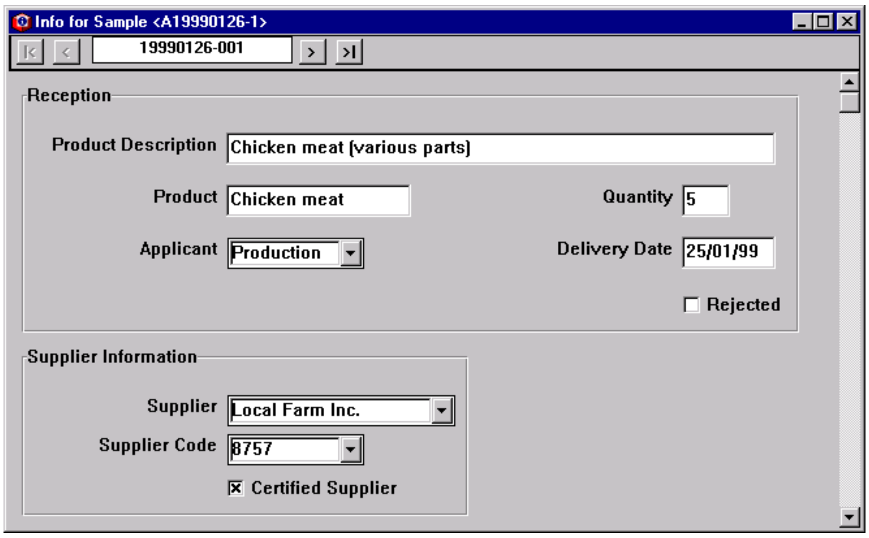
However, infocard and/or test plan assignment can also be controlled by a number of preferences (**scCreateIc** for info card assignment and **scCreatePg** for test plan assignment).

4 Info Cards

4.1 Introduction

An info card is a group of info fields that appear together on the screen in a particular configuration. The grouping is done on a logical 'belonging together' basis, not on grounds of being assigned to a specific sample. Info cards allow for the entry of sample information that is not related to the laboratory tests.

The picture below shows an example of info cards **Reception and Supplier information** for sample 19990126-001.



Reception

Product Description: Chicken meat (various parts)

Product: Chicken meat

Quantity: 5

Applicant: Production

Delivery Date: 25/01/99

☐ Rejected

Supplier Information

Supplier: Local Farm Inc.

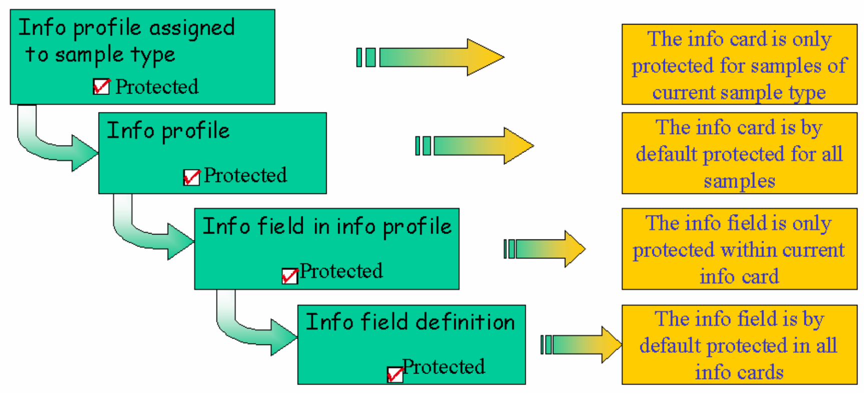
Supplier Code: 8757

☒ Certified Supplier

The definition of an info card is done through an info profile. Each sample type may have more than one info profile assigned. Each info profile can be assigned to several sample types.

Many settings for info fields and info cards can be controlled at different levels. These settings demonstrate hierarchical behaviour. The choice where the property will be set depends upon the scope of the required functionality. For instance take the protected flag. When an info field should always be protected, this property must be defined at the info-field definition level. For info fields that should only be protected within the context of specific info cards, the **Protected** flag can be set on the link "info profile-info field". This setting actually overrules the setting at the info-field definition level. When the info card itself should be entirely protected, the flag can be set at the info profile level. Finally, the setting can be overruled at "sample type-info profile" level, when the protected property of the info card is sample-type dependent.

The picture below shows the hierarchical setting of info-field behaviour.


Info profile assigned to sample type
Protected
Info profile
Protected
Info field in info profile
Protected
Info field definition
Protected
The info card is only protected for samples of current sample type
The info card is by default protected for all samples
The info field is only protected within current info card
The info field is by default protected in all info cards

4.2 Info-field Properties

Info fields are the building blocks for info cards. Although a defined info field can be assigned to many different info profiles, the definition of the info field itself is centralized. This guarantees a consistent format for both the info field itself and for its data input.

Example: In an info field **Batch number**, the batch number has always a length of 20 characters and the list of valid numbers needs to be defined only once. This eliminates/reduces user typing errors and makes queries on all samples belonging to the same batch simpler.

4.2.1 Info-field Standard Properties

Each info-field definition is uniquely identified by a name. This name is normally only used by the application manager (as an internal technical reference) and is not displayed at the operational level. The info-field title displayed within the info cards at the operational level is set through the display title in the info-field display properties.

For individual info fields at the operational level, no history is kept. Info fields at the operational level have by default no life cycle and no access rights. Whether or not info fields are active or modifiable is entirely controlled at the info card level (through the info card life cycle). When the info card is active and modifiable, so are all its info fields. In addition, access to info fields is completely controlled through the access rights at the info card level.

Important

In principle, it is possible to define a specific life cycle for a specific info field. This is strongly discouraged because of the huge load it generates (in terms of events that are generated. This option should only be used in very exceptional situations – when there is no other alternative).

4.2.2 Validation Function

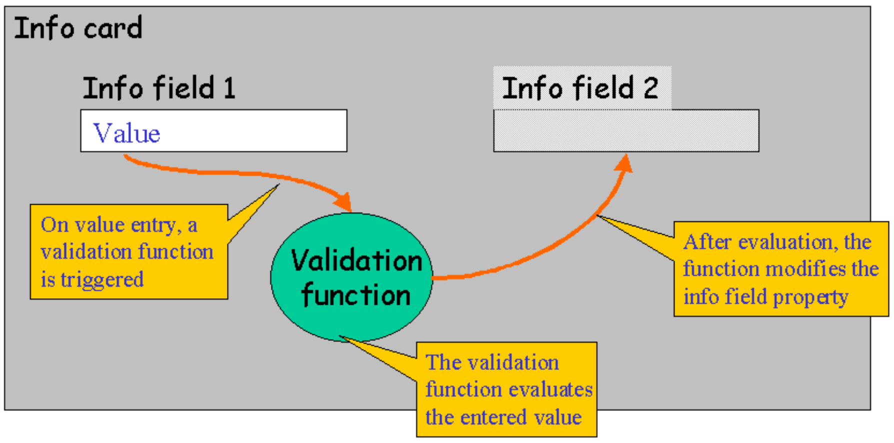
For an info field, a custom validation function can be defined. This function, which is developed in Visual C++, is used to determine whether or not a specific value entered in the info field will be accepted as a valid value.

The custom validation function is triggered whenever the field is edited/changed. How the value was entered (through the keyboard, pasting from the clipboard or requesting the default value) does not make any difference. This validation is triggered when the field control loses the focus (i.e. the editing action is complete).

The validation function has access to the full list of all info fields (which occur in the assigned info profile(s)/info card(s)). Within the validation function, any property of any info field can be modified (although often nothing is modified because only the contents are verified). Therefore, validation functions can be used to reset the properties of other info fields in the info card, based upon the value entered into a specific info field.

The validation function can be called by the Client Event Manager when it receives an event (e.g. from another field).

The picture below shows the concept of dynamic modification of info-field properties using a validation function.


Info card
Info field 1
Value
Info field 2
On value entry, a validation function is triggered
Validation function
After evaluation, the function modifies the info field property
The validation function evaluates the entered value

Another example of the use of info-field validation functions is the assignment of a specific test plan based on the value filled out in an info field (for instance, a customer-specific test plan assigned based on the name of a customer filled out in an info field).

4.2.3 Hidden, Protected and Mandatory Info Fields

Info fields in an info card can be:

- Hidden
- Mandatory
- Protected

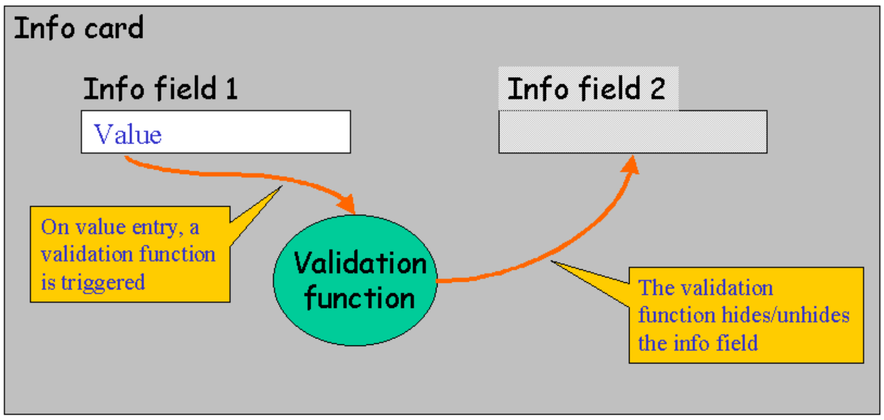
Each of these properties can be set at the info-field definition level. Once the info field has been assigned to an info profile, these properties can be overruled (on the “info profile-info field” link). In this manner, an info field that is by default not mandatory can be made mandatory in a specific info card.

Each of these properties can also be set dynamically at the operational level. This dynamic setting is done by a validation function that evaluates the contents of another info field. An info field can for instance be hidden based on the value filled out in another info field.

Hidden info fields

It is possible to dynamically hide (or show) an info field by changing the value of its **Hidden** flag through a custom function. In this manner, certain information can be hidden or revealed.

The picture below shows the concept of dynamic hiding/showing of info fields.


Info card
Info field 1
Value
Info field 2
On value entry, a validation function is triggered
Validation function
The validation function hides/unhides the info field

As soon as the value for info field 1 is filled out, a custom validation function defined for info field 1 is triggered. This validation function evaluates the contents of the info field. The validation function has full access to the properties of all info fields in the info card. In this way, the **Hidden** property for another info field can be switched on/off.

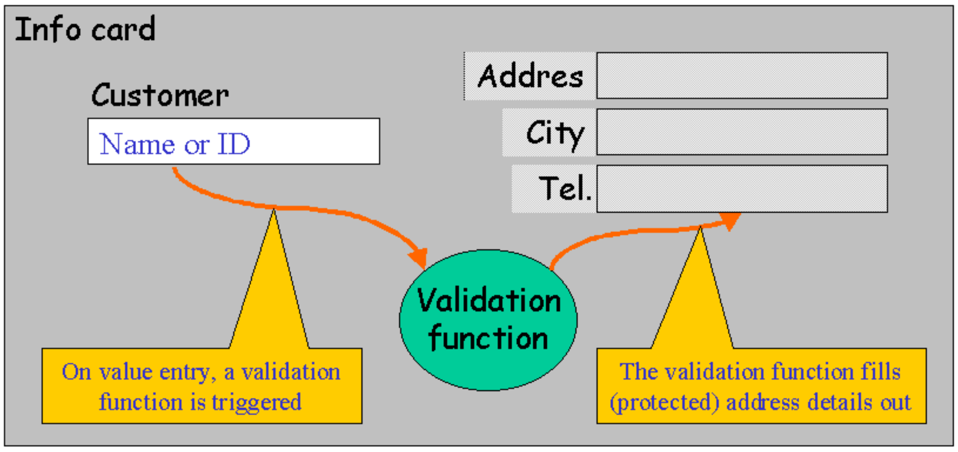
In the same manner, custom validation logic can be used to show/hide an entire info card. Likewise, in some specific cases, info fields can be dynamically removed or attached to an info card at the operational level as well.

Protected info fields

Info fields, for which a default value has been defined, can be protected, which means the user cannot modify the value contained in the info field.

Protecting info fields is especially useful when the field contents is controlled by a custom function. An example is the automatic generation of the address details based on a customer’s name filled out in an info field.

The picture below shows the concept of dynamically updating protected info fields.


Info card
Customer
Name or ID
Addres
City
Tel.
On value entry, a validation function is triggered
Validation function
The validation function fills (protected) address details out

Mandatory info fields

Info fields requiring data entry can be set as mandatory. A user cannot close an info card without supplying values for mandatory info fields.

4.2.4 Info-field Data Properties

For each info field, the data type and format can be specified. The table below lists the available data types.

Type	Description
Text	Alphanumeric characters
Float Number	Number with decimal
Integer Number	Number without decimals
Date	Date/Time

An info field can also accommodate for long text results.

Tip

For more info on data types and formatting, read the chapter on the formatting of results in the **Unilab Concepts Guide part 3**.

4.2.5 Default Value

The user can specify a default value for an info field when it is used in an info card (at the operational level). These default values can be overruled when assigning the info field to an info profile. Defaults can be defined as fixed values, sample properties or attributes of another object.

Fixed values (hard-coded defaults)

The user can enter any fixed text as a default value for an operational info field. This value will automatically appear as the default value for the info field in a newly created sample. Some restrictions apply to the use of fixed default values. The value must be of the same type as the info-field data type and will be formatted according to the defined info field.

Sample (request) properties

The actual sample (request) property value (at run time) will be used as a default value for the info field. This default actually links the info field to a specific standard sample property. Some standard sample properties are read-only; others can be changed but only within certain limits.

Changing the info-field value automatically changes the standard sample property too. Unilab tries to enforce this restriction automatically.

The table below lists the existing sample/request standard properties and indicates the limitations imposed on info fields that use these properties as a default value.

Important

Some fields are only supported for samples and others are only supported for requests. When defining any field, it is impossible to know if it will be used for a request or a sample, therefore the system cannot impose this restriction automatically.

The table below lists the info-field default restrictions for sample/request standard properties.

Property	**Sc Rq**	**Datatype**	**Other requirements**
Allow any pp	☑☐	Integer	Checkbox
Allow any sc	☐☑	Integer	Checkbox
Allow any st	☐☑	Integer	Checkbox
Created_by	☑☑	Text	
Creation_date	☑☑	Date	
Date 1	☑☑	Date	
Date 2	☑☑	Date	
Date 3	☑☑	Date	
Date 4	☑☑	Date	
Date 5	☑☑	Date	
Description	☑☑	Text	
Due Date	☐☑	Date	
Exec_end_date	☑☑	Date	
Exec_start_date	☑☑	Date	
Label Format	☑☑	Text	
Priority	☑☑	Integer	
Request	☑☑	Text	Always protected (read-only)
Request Type	☐☑	Text	
Sample code	☑☑	Text	Always protected (read-only)
Sample type	☑☐	Text	
Sampling Date	☑☑	Date	
Shelf Life Limit	☑☐	Integer	
Shelf Life Unit	☑☐	Text	
SOP	☑☑	Text	
Responsible	☑☑	Text	

Attribute values

Attribute values can be used as default value for info fields as well. The info field will be initialized with the value of an attribute assigned to the sample, the sample type or the request to which the sample belongs. This default actually links the info field to a specific attribute.

Care must be taken that the data values in the attribute match the type and the format of the info field. Unilab does not check this restriction, thus the user must take care of it.

4.2.6 Info-field Display Properties

The representation of an info field on the screen (within an info profile or an info card) can be set through the following properties:

- A screen display title to be shown in front of the info field
- The size of the info field
- The kind of control to be used on the screen to fill in the info value (the display type).

Size of an info field

The size (= the width and the height) of the info field is expressed relative to the average size of a character (which depends on the font that will be used). To make sure that the info field is displayed in the same way on every info profile where it is used, the width and the height are included in the info-field properties. Info fields cannot be resized within info profiles.

Display type

The following info-field display types are supported:

- An input edit field
- A list box
- A drop-down list (supplying a value not in the list is forbidden)
- A combo box (entering a value which is not in the list is allowed)
- A (tri-state) check box, indicating a Boolean value (stored as 1 for checked and 0 for not checked, when grayed, a value of 2 is stored in the database)
- A linked document field with an attribute service attached to it used to store documents or to launch other Microsoft Windows applications (e.g. Microsoft Word, Microsoft Excel, etc.). The contents of the field can refer to a file in a network accessible file system.
- Fixed text
- Date control
- Edit mask
- Push button
- Radio button
- Spin box

Additional property details depend on the specified display type.

For list boxes, drop down list boxes and combo boxes, a list of possible values can be defined as a fixed or a dynamic list (by means of an SQL statement in the database).

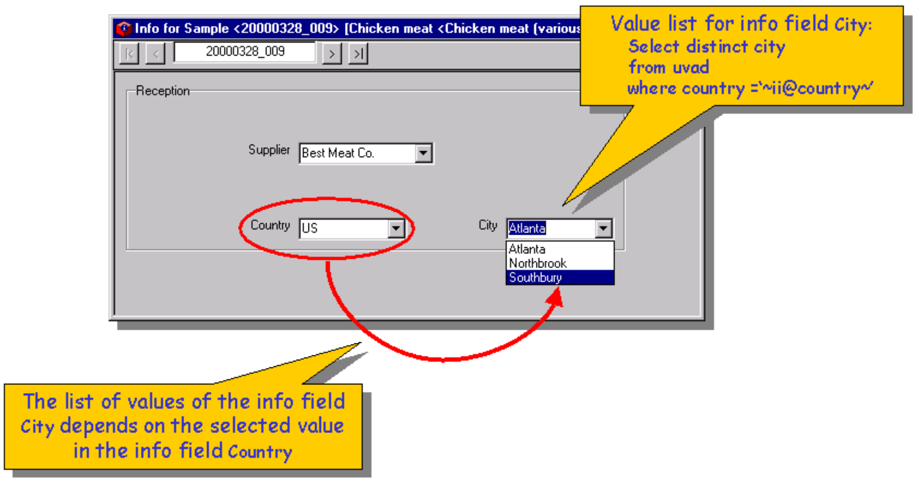
Both fixed and dynamic SQL statements can be used. For example, dynamic SQL statements allow you to make a list of values based on a sample property or on another info-field value. Within the SQL command, tilde substitution can be used. The following syntax is used to dynamically retrieve the correct list of values. A complete and up–o-date overview on tilde substitution is given in the document **SIMATIC IT Unilab – Customizing the system** in the Unilab Product Library.

Value	Description
~ii@<ii>~	Substitutes the actual value of the specified info field <ii>
~sc@<std **prop>~**	Substitutes the actual value of the specified standard property of the current sample
~scgk@<gk>~	Substitutes the actual value of the specified sample group key <gk>.
~scau@<au>~	Substitutes the actual value of the specified sample attribute <au>.
~rq@<std **prop>~**	Substitutes the actual value of the specified standard property of the current request
~rqgk@<gk>~	Substitutes the actual value of the specified request group key <gk>.
~rqau@<au>~	Substitutes the actual value of the specified request attribute <au>.

Example: The following SQL-command will return the list of cities for the currently-selected country (which is just another info field within the same info card).

```
SELECT DISTINCT city
FROM uvad
WHERE country = '~ii@country~'
```

The picture below shows the dynamic value list for an info field.


Info for Sample <20000328_009> [Chicken meat <Chicken meat (various

20000328_009

Reception

Supplier: Best Meat Co.

Country: US

City: Atlanta

Atlanta
Northbrook
Southbury

Value list for info field City:
Select distinct city
from uvad
where country ='~ii@country~'
The list of values of the info field City depends on the selected value in the info field Country

4.2.7 Restrictions on Info-field Definitions

In order to guarantee proper system operation, a number of constraints are applicable. Therefore, the following checks are performed before an info-field definition can be saved to the database (in the same order as they are specified here):

- If the info field has a display type **Checkbox**, the data type must be an Integer and the format must be "I"
- If the info field has a display type **Checkbox**, and the default value is **Fixed**, the default value must be either **0** or **1**
- If the info field has a display type **Checkbox** and the default value is a sample attribute, then the attribute type must be a Boolean (e.g**. Allow any pp**)
- If the default value is a sample attribute then the data type and display type must be of the types specified in the table mentioned above.

Note

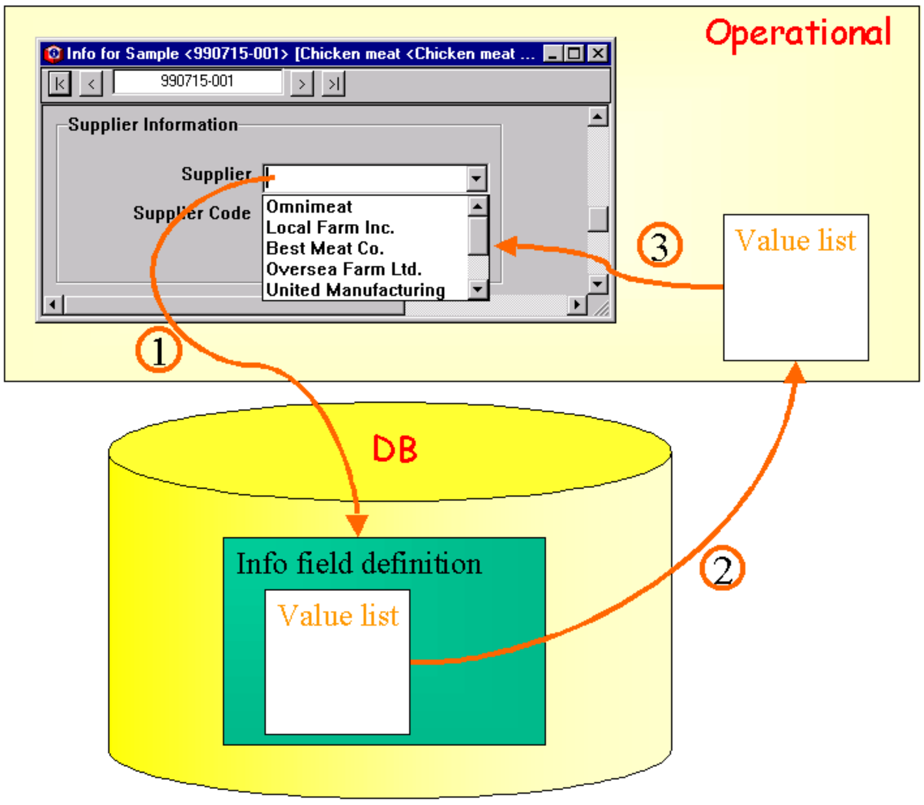
There are no additional checks when the default value is a user-defined attribute.

4.2.8 Info Fields at Operational Level

At the operational level, info fields are created based on their configuration level definitions. However, not all info-field definition details are copied from the configuration to operational level. For example, this is the case for the custom validation function, the data type and format and the list of values for list boxes. The main reason for this approach is the fact that copying all these details to the operational level does not provide any added value, while consuming a lot of database storage. Moreover, this mechanism provides a more flexible and dynamic update of value lists.

Each time this information is required at the operational level, it is retrieved from configuration. This concept is illustrated in the figure below. When opening a drop-down list box at the operational level (1), the list of values is fetched from the info-field definition at the configuration level (2) and cached into client memory. Based on this value list, the drop-down list box is filled for display.

The figure below shows how info-field details are fetched from configuration level.



This mechanism implies that the appropriate properties cannot be obtained if the info-field definition is missing. In that case, the info field will appear disabled (grayed) in the info card.

Note that the info-fields audit trail is logged on info fields when a lifecycle is defined on this level. When no lifecycle is present for info fields, the audit trail is logged on info card level.

4.3 Info Profile Properties

The definition of an Info Profile consists of:

- The layout containing all assigned info fields (including (x, y) position and possible default value)
- Standard attributes, including name, description, a second description (for reporting purposes) and an indication that the corresponding info card will always be hidden and/or protected.

4.3.1 Info Fields in an Info Profile

Within each info profile, an infinite number of info fields can be added and positioned. Both active and inactive info fields can be used. Each field on the profile gets some extra properties.

Positioning and sizing of info fields

Info fields on the info profile can be selected and moved around using standard Microsoft Windows selection mechanisms. The X and Y position of the info field's lower left corner in the screen layout can be set for each info field in the info profile. This is useful for aligning a number of fields exactly with one another.

Resizing info fields in an info profile is not allowed thus ensuring that info fields appear exactly the same in all profiles. In other words: the **Width** and **Height** of the info field will be used as defined in the info-field definition itself.

Tab order of info fields

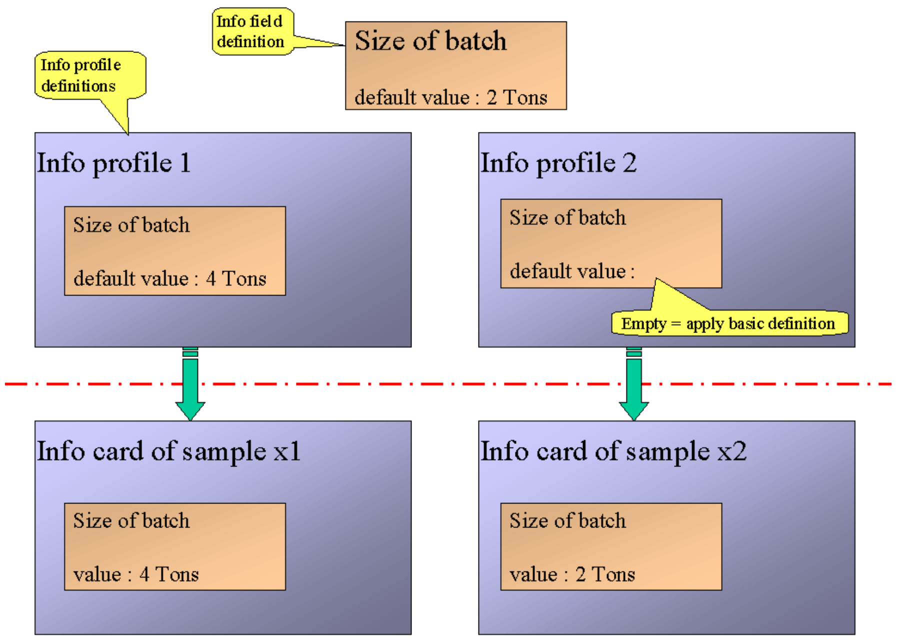
The order in which the info fields are filled out in an info card is often important. Setting the tab order for the info fields in an info profile can control the order in which the info fields are accessed when clicking the **[Tab]** button.

Overruling info-field properties

Some of the info-field specific properties can be changed within the context of a specific info profile. For each info field, the **Hidden**, **Mandatory** and **Protected** flags can be overruled once it is assigned to an info profile. In this way an info field that is not mandatory by default, can be set to mandatory when applied to a specific info card.

For each info field, the default value from the info-field definition can also be overruled within the context of a specific info profile.

The picture below shows how info-field properties are overruled at info profile level.



For checkboxes (which are by definition either in the **unchecked = 0** or **checked = 1** state), a special value is added to indicate that the default value of the info field must be used. Therefore, checkboxes in an info profile are tri-state checkboxes: when **grayed = 2** the default value of the info-field definition is applied when the info card is created.

4.3.2 Info Cards on Operational Level

At sample creation, info cards are assigned according to the frequency set at the configuration level.

Tip

A number of preferences control the actual creation of the info cards at sample login.

Info cards can also be dynamically assigned or removed from samples at the operational level. However, the user must have the appropriate functional access rights to do so.

Data access rights on info cards control who is allowed to consult or modify the values contained in the info card fields. Users belonging to user profiles, for which the access rights are set to **None**, will not even know that the info cards exist.

Keep in mind that an object (method cell, method, parameter,....) can be modified when one of its parents is part of a data-domain where the currently logged-on user has only Read-Only privileges if the system setting CASCADE_READONLY is set to NO. However, if one of the parents has 'No Access' rights to this data domain, it remains impossible to modify the object.

For more information on data access rights, see par. Data Access Rights in the **SIMATIC IT Unilab -Concepts Guide Part 3**.

Important

Users having **Write** access to an info card might not be able to fill in or modify info-field values: this depends on the setting of the protected flags for individual info-fields or for the entire info card.

5 Parameter Groups and Specifications

5.1 Introduction

A parameter group is a group of related parameters, assigned to a sample. The relationship between these parameters can be of a different nature. For instance, parameters can be grouped based on the fact that they are always measured together or analyzed in the same section of the laboratory. Grouping of parameters can also be based on a similar characteristic, e.g. grouping all chemical, microbiological, and sensorial analyses. Parameter groups are defined at the configuration level by means of a parameter profile.

For each individual parameter in a parameter group, a number of specifications can be defined. These specifications may have different values depending on the parameter group to which the parameter belongs.

5.2 Parameter-group Properties

The majority of parameter-group properties are copied from its parameter profile at the configuration level. However, some properties are specifically managed at the parameter-group level as well.

5.2.1 Processing Details of a Parameter Group

Parameter-group processing details are first provided by its current status in its life cycle. The default life cycle model used for newly created parameter groups is set at the parameter profile level. Authorized users can always assign another life cycle model to the parameter group.

The date that the parameter group was created and the responsible person for the parameter-group creation are also kept in the parameter-group properties. For parameter groups that have been assigned manually, the assignment information is included.

The start of the analyses (entering parameter results) and the eventual completion of the analyses are also supplied automatically.

In case re-analyses have been performed for a parameter group, the number of re-analyses is automatically filled out in the parameter-group properties.

5.2.2 Parameter-group Result

In some cases, a result is required for an entire parameter group. For example, laboratories doing sensorial analyses (= parameter group) will not include each parameter detail in a certificate that is sent to a customer. Instead, the certificate will hold a statement that the sensorial analyses are **OK**.

For parameter-group results, a format and a unit can be specified at the parameter profile level. These settings are copied to the operational level and can be modified if required.

Tip

For more information on data types and formatting, refer to the chapter on Formatting of results.

The value itself is entered and stored at the parameter-group level. The value can either be entered manually or through an action in the parameter-group life cycle. Unilab also records whether the value was entered manually or obtained automatically through an algorithm. For parameter-group results, long text results are supported as well.

5.2.3 Cancelling Parameter Groups

Parameter groups can be cancelled. Cancelling a parameter group automatically cancels all parameters and corresponding methods as well. For cancelled methods, the worklist assignment rules are then evaluated. For each cancelled object, the execution end date is supplied. If the execution start date was still empty, this date is also filled out.

Note that parameters and methods that are not modifiable will not be cancelled.

5.2.4 Reanalyzing Parameter Groups

When a parameter group is re-analyzed (as a whole), all descending objects (parameters and methods) will also be re-analyzed. For methods, worklist assignment rules are re-evaluated and the methods are automatically inserted into the correct worklist.

The status the parameter group receives on re-analysis depends on the actual life cycle definition.

5.2.5 Assignment of Parameter Groups

Manual or automatic assignment

Parameter groups can either be assigned automatically or manually for created samples. The automatic assignment of parameter groups is controlled through the preference **scCreatePg**. This preference specifies whether the parameter groups are assigned:

- On sample creation
- Once an info card has been filled out (info-field value dependent test plan)
- Only manual (not automatic).

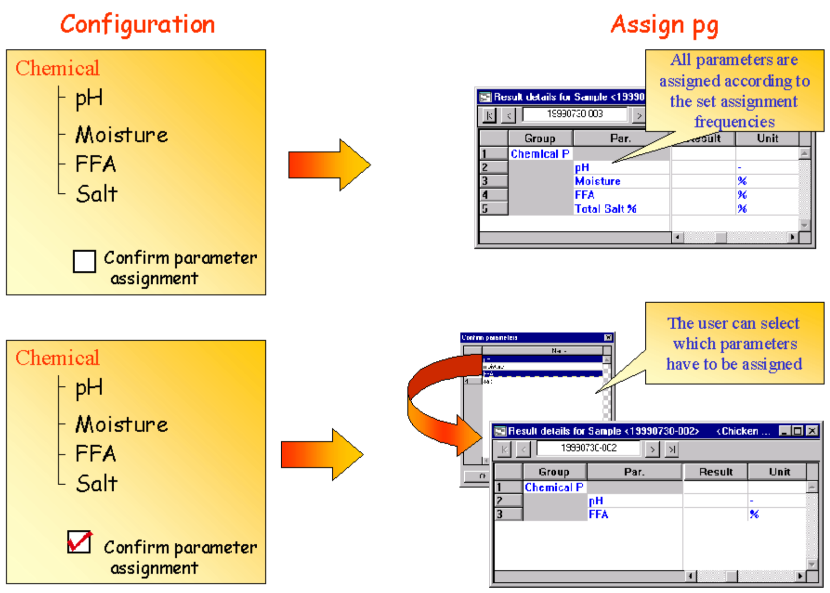
Additional parameter groups can always be assigned manually. The parameter-groups list from which the user can choose depends on the sample-type configuration settings. In the sample-type properties it can be set whether the list contains all configured parameter profiles, or only those that were explicitly assigned to the sample type.

Information related to the assignment of parameter groups is kept in their standard properties. This information includes the assignment date and the user who assigned the parameter group.

Confirm parameter assignment

On manual assignment, the user has the option to confirm the assignment of all parameters belonging to the profile. This behaviour can be implemented by setting the **Confirm parameter assignment** flag at the parameter profile level. When the user manually assigns the corresponding parameter group to a sample, he/she will get the pick list of all parameters within the parameter profile. The user can then selectively assign the required parameters.

The picture below shows how to confirm parameter assignment.


Configuration
Assign pg
Chemical
pH
Moisture
FFA
Salt
Confirm parameter assignment
All parameters are assigned according to the set assignment frequencies
Result details for Sample <19990
19990730 003
Chemical
pH
Moisture
FFA
Salt
Confirm parameter assignment
The user can select which parameters have to be assigned
Confirm parameters
Result details for Sample <19990730-002> <Chicken ...
1999073C-0C2

	Group	Par.	Result	Unit
1	Chemical P			
2		pH		-
3		Moisture		%
4		FFA		%
5		Total Salt %		%

	Group	Par.	Result	Unit
1	Chemical P			
2		pH		-
3		FFA		%

Overruling the assignment frequencies

The **Assign full test plan** option allows overruling the configured assignment frequencies. All parameter groups that were not assigned according to the configured assignment frequencies to the current sample can be assigned manually.

5.3 Parameter Profiles

The properties and the behaviour of a parameter group are defined at the configuration level by means of a parameter profile.

5.3.1 Parameter Profile Properties

The properties of the parameter profile itself conform to the generic object properties.

Most of the parameter profile standard properties are used to control the behaviour of the parameter group at the operational level. This is the case for the life cycle applied to parameter groups of the current parameter profile, the format of the parameter-group result, the delay on the parameter group and the flags used to control parameter assignment at the operational level.

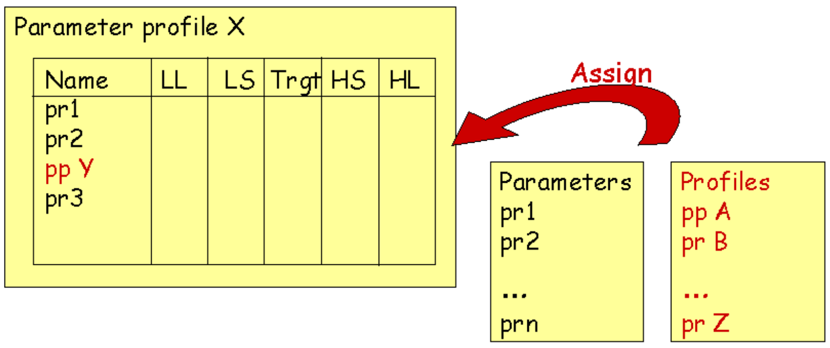
5.3.2 Parameter Profile Component Concept

A parameter profile contains the lists of components to be assigned to the parameter group on the operational level and their assignment frequencies. These components can both be parameter definitions or other parameter profiles.

By default, a parameter-group groups a set of related parameters. That is why the component list of a parameter profile consists in many cases of a list of parameter definitions.

However, a number of practical situations may require that other parameter profiles are included in the list. This may simplify the configuration and assignment of test plans where a set of analyses is to be considered as one item within a parameter profile.

The picture below shows the parameter profile component concept.



Note that the hierarchical relations on the operational level do not change if a parameter profile is assigned to another parameter profile. These nested parameter profiles are expanded to normal parameter groups on the operational level.

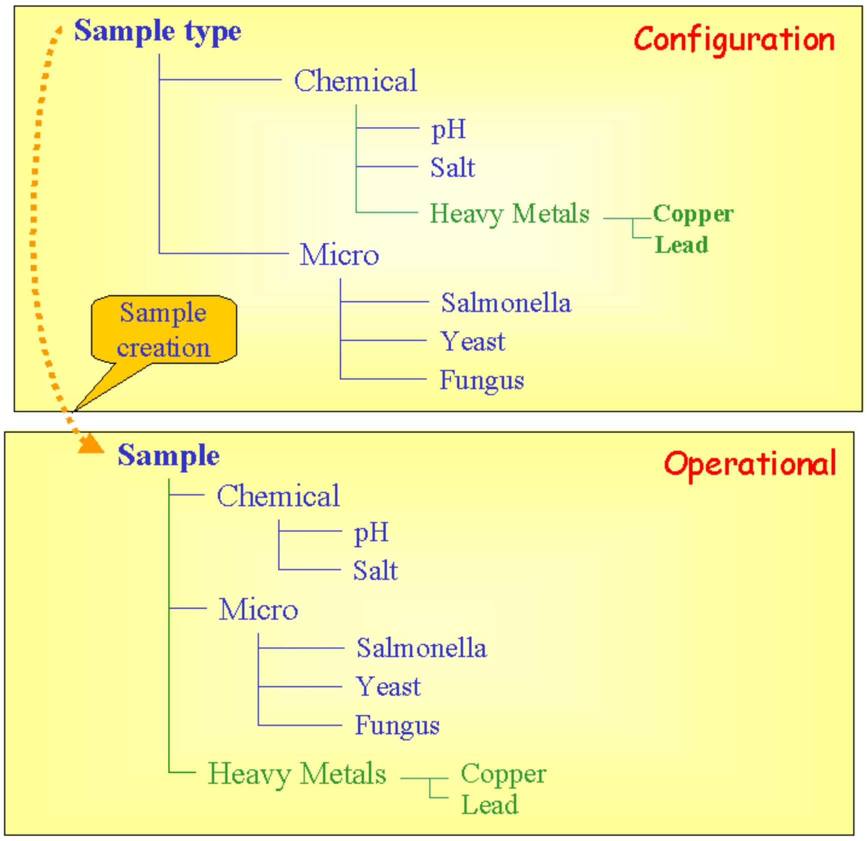
In principle, it is possible that a parameter profile contained in another parameter profile, may, in turn, contain another parameter profile. Suppose then that, for all PPs, the ‘confirm assign pa’ flag is set and that, at a certain time, PP X must be assigned. The ‘Assign pg’ functionality will then only allow confirming the assignment of PP Y, PR A and PR B, i.e. PP Y is really treated as a parameter. It will not be possible to further specify whether e.g. PR Z, PR C or PR D must be assigned. The same idea is valid for the **Assign Full Test Plan** functionality.

Example: In a service laboratory, the analyses request forms mention different sets of analyses that can be executed on demand. This list may include chemical analyses (pH, salt,), microbiological analyses (Salmonella, Yeast) and others. However, some of the items mentioned in the parameter group Chemical may consist of a package of analyses. The analysis Heavy Metals, for instance, involves the detection of a number of parameters (Cu, Pb, Zn,) in a sample.

All of these parameters are combined in one parameter profile called **Heavy Metals**. Subsequently, this profile can then be added to the component list of the parameter profile **Chemical**.

Upon sample login, the analyst can select the profile **Heavy metals** from the test plan. This component will then be expanded to an additional parameter group assigned to the sample.

The picture below shows the parameter profile as a component of a parameter profile.



5.3.3 Properties of Components Assigned to Parameter Profile

A list of parameter definitions can be assigned to the parameter profile. For each of these parameters, a number of properties can be defined. Some of these properties overrule the actual parameter definition properties themselves. The properties that can be defined on the link “parameter profile–parameter” include:

- Parameter-specific properties
- Parameter profile/parameter definition/method bridge
- Assignment frequency
- Specifications
- User-definable attributes

Parameter-specific properties

Certain properties that are specific for a parameter can be set on the link “parameter profile–parameter”. This includes properties that overrule the actual parameter definition properties such as the delay. Delays specified at the parameter level add up with delays specified at the parameter-group level. It might therefore be necessary to overrule the delay specified at the parameter level in case the parameter is assigned to a parameter profile that has a delay itself.

For the assigned parameter, the number of measurements can also be defined and whether or not additional assignments are allowed.

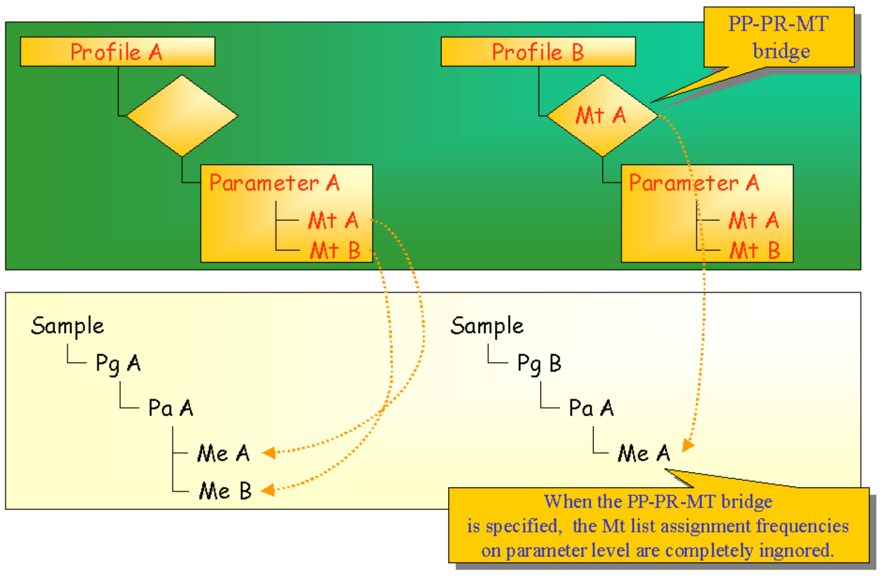
Parameter profile/parameter definition/method bridge

For each parameter in a parameter profile, a number of methods can be defined. However, in a number of cases the information on *how to proceed* to obtain test information (= which test method to use) is not linked directly to the parameter definition itself, but rather to the parameter profile.

Water quality control can illustrate the need of this information at the parameter profile level. Some of the parameters measured in the context of waste water quality are identical to those used for the assessment of drinking water quality. Nevertheless, the ranges of the measured concentrations, as well as the specifications, are obviously significantly different. In fact, completely different measurement methods are applied. The differences are captured by making use of different parameter profiles: one for waste water, another one for drinking water. It is possible in Unilab to associate a parameter in a parameter profile not only with a frequency and specifications, but also with a specific test method (including the number of times the method needs to be executed). This link between the parameter profile, the parameter, and the method is called the PP-PR-MT bridge (**parameter profile/parameter definition/method** bridge).

If a method assignment rule is configured within a parameter profile, the method assignment configuration at the parameter definition level is ignored.

The figure below illustrates the concept of a PP-PR-MT bridge.


Profile A
Profile B
PP-PR-MT bridge
Mt A
Parameter A
Mt A
Mt B
Parameter A
Mt A
Mt B
Sample
Pg A
Pa A
Me A
Me B
Sample
Pg B
Pa A
Me A
When the PP-PR-MT bridge is specified, the Mt list assignment frequencies on parameter level are completely ingnored.

The PP-PR-MT bridge also allows assigning the same parameter twice to the same parameter profile with another method. However, the specifications cannot be different (for one and the same parameter in a single parameter profile).

Assignment frequency

The automatic assignment of parameters to a parameter group is controlled through assignment frequencies at the configuration level.

The frequency only applies at the level “parameter group–parameter” and is independent of the sample type. The assignment frequency can however be made sample-type dependent by setting the appropriate flag.

Tip

For more info on assignment frequencies, refer to the chapter on frequency filtering.

At the operational level, parameters can be assigned ad hoc to the parameter group. The selection list displayed at the operational level is fetched from the configuration level. In the parameter profile, it can be set whether this list should be restricted to the parameters that have been explicitly assigned to the parameter-group definition or not. If not, the analyst sees the entire list of active parameter definitions from which to choose.

Specifications

The specifications used for result validation are included in the parameter definition itself, and are set on the link “parameter profile–parameter”. This allows for more flexibility in the parameter definition: one parameter can be assigned to multiple parameter profiles, each time with different specifications.

User-definable attributes

Some user-defined attributes can be assigned to the link “parameter profile–parameter”. An example is a flag that indicates which parameters are critical to production (to automatically release/validate the parameter group).

Note that there is a limitation to the attributes at the “parameter profile-parameter” link level. Multiple assignments of the same parameter to the same parameter profile always have the same list of attributes and attribute values. It is not possible to assign a different attribute list for multiple instances of the same parameter within the same parameter profile. This is due to the fact that for technical reasons (mainly involved in re-sequencing or re-sorting of the parameter list), the sequences used to uniquely identify the “parameter profile–parameter” relationship are not copied into the table where the attributes on the “parameter profile-parameter” link are stored.

5.3.4 Delay of Parameter Group

On certain parameter groups, a delay needs to be specified. An example is provided by shelf-life tests. Shelf-life tests include a number of parameters that are analyzed on the same sample within specified time intervals (for instance after one week, after two weeks, etc.).

All analysis for which the same delay is applied can be grouped into one parameter group. For that parameter group, this delay can be set. This delay specifies how long after the sample sampling date the parameter group should become active. The initial state of parameter groups for which a delay has been specified will be **“@D – Delayed** normally the objects in this state are neither active nor modifiable. The elapse of the delay triggers a state transition to the next/initial state of the specified life cycle.

Example: Shelf-life testing implies the repetition of a number of analyses (e.g. growth of certain bacteria) within a set interval (e.g. every week) on the same sample. A number of parameter profiles can be configured, called e.g. **Analyses_1_week**, **Analyses_2_weeks**, … , for which the delay is set to **1 week**, **2 weeks**, etc. The corresponding parameter groups' life cycle contains an initial state **Delayed** in which the parameter groups are not active and not modifiable. Only after the elapse of the delay, a state transition is triggered to status **Available** (thus respectively after 1 week, 2 weeks, etc.).

Note

A delay can also be specified at the parameter level. Note that delays at the parameter and parameter-group level add up: a parameter delay is only counted from the moment that the parameter group becomes active.

5.4 Specifications

5.4.1 Specification Definitions

A set of specifications can be associated with each parameter within a parameter profile. These specifications are used for validation purposes or to trigger certain events in case the result is outside the specifications.

The alarms that should be triggered in case a parameter result lies outside the specifications are defined at the parameter level.

5.4.2 Specification Sets

Three sets of specifications can be configured in Unilab. The three sets can be named by the application manager through the appropriate system settings. By default, these specification sets have the following names:

- Product specifications
- Production specifications
- Legal specifications.

Most of the time, there is just one specification set for each parameter.

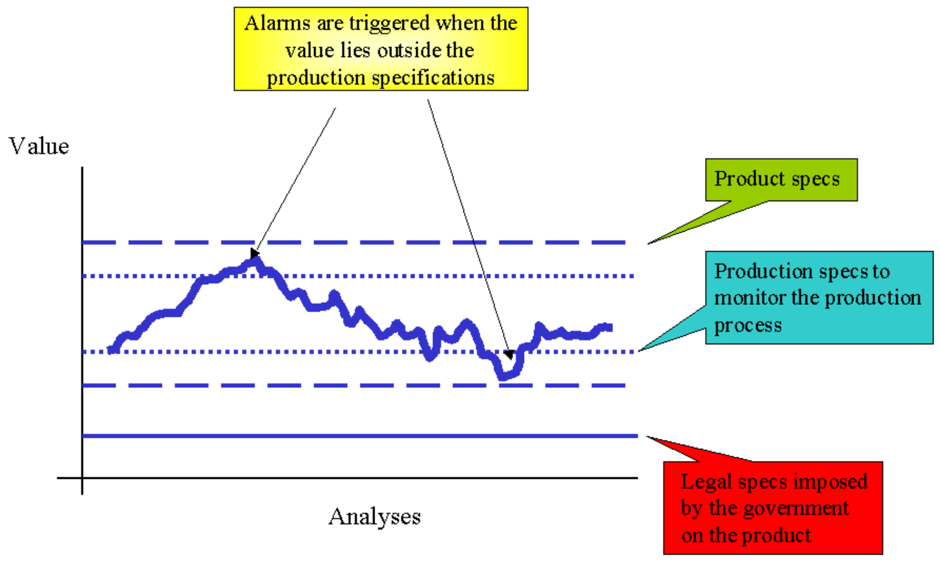
Note

The labels for the specification sets can be defined through the system settings **LABEL_SETA**, **LABEL_SETB** and **LABEL_SETC**.

The order in which the specification sets are evaluated can be defined per parameter.

Customer- or supplier-specific specifications cannot be handled by using additional specification sets; there are usually (many) more than three suppliers/customers of a particular product. Customer- and supplier-specific specifications are handled using the concept of multiple test plans.

The picture below shows the use of three specification sets in a manufacturing environment.


Alarms are triggered when the value lies outside the production specifications
Value
Product specs
Production specs to monitor the production process
Legal specs imposed by the government on the product
Analyses

A chocolate-producing company uses three sets of specifications. The law dictates that the product should at least contain 60% fat (= legal specifications). The company itself produces a product that has a fat content of 70 -80 % (= product specifications). Internally, they apply specifications of 72 – 78% to monitor the chocolate production (= production specifications).

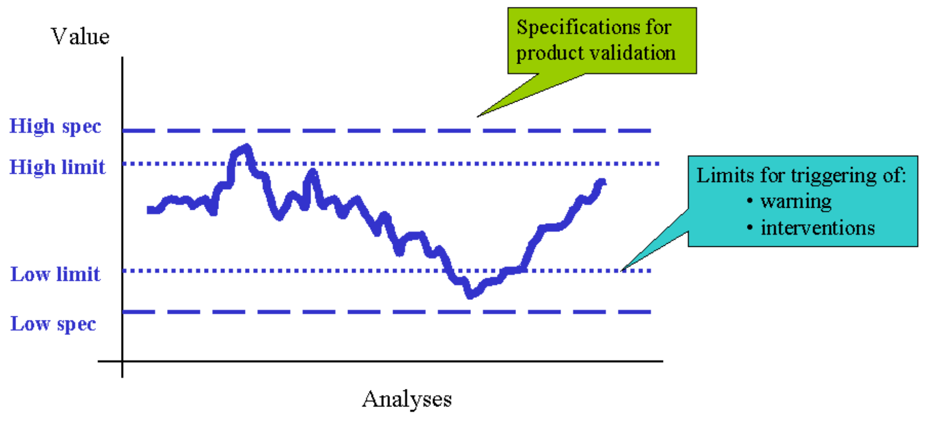
Each of these specification sets has the following (detailed) structure:

- Low and high spec (indicated as LS and HS)
- Low and high limit (indicated as LL and HL)
- Target and allowed low and high deviation (indicated as Target, LTD and HTD).

The deviation can be expressed as an absolute value (relative to the target) or as a percentage of the target value.

The use and the value for each of these specifications are subject to customer interpretation. By default, the low and high specifications are used for sample validation and subsequent product release. A value outside the low limit or the high limit can indicate that the production process is deviating. The limits can also be used as a warning to avoid a value outside the low and high specifications, and thus to avoid product rejection. The target value and its deviation can also be applied to trigger specific interventions.

The picture below shows the use of specs and limits in a manufacturing environment.



Note

The laboratory does not have to implement all specification sets; neither does it need to use all information in a specification set. In some manufacturing environments, no legal specifications are applied. Laboratories doing environmental analyses often only deal with legal specifications.

5.4.3 Different test plans depending on different specified criteria

In several cases, specifications depend not only on the parameter and the parameter profile, but on other items such as the sample type (product), the customer for which the products are made, or even the plant where the product is manufactured. As a matter of fact, on some occasions, completely different test plans need to be executed depending on criteria such as production plant, customer, product, etc..

To support these environments, the primary parameter profile key is extended with 5 extra key fields that allows configurators to define a hierarchy on the parameter-profile level. These key fields can be defined freely according to the specific needs and environment of the enterprise context. By this, a specific corporate structure with, for instance, different plants, products and suppliers can be mapped to a hierarchical parameter profile structure, permitting the existence of different test plans, depending on the ‘context’ of a specific sample.

This concept is clearly in line with the undeniable and growing tendency across all industries to shift from departmental to Centralized Quality Management deployments. As the concept of full CQM support spans a vast amount of different requirements like support for multi-testplan, multi-timezone, multi-language, and multiple database instances, a separate manual has been made to describe and explain the standard functionality to enable Centralized Quality Management support.

For more information on how to set-up a configuration supporting the concept of multiple test plans, or CQM in general, the document ‘Centralized Quality Management’ can be consulted in the Unilab Product Library.

6 Parameters

Parameters are sample properties that can be measured. They provide the answer to the question *"What must be analyzed?"*. The parameter result corresponds in most cases to the analysis result that will be reported. All parameters that logically belong together can be grouped in a parameter profile that can be assigned to a sample type or directly to a sample.

A parameter can be determined using one or more methods. The methods determine *how* the analysis must be done. During configuration, all possible methods for a parameter are to be indicated. At run-time, a certain method (or several) may be chosen from the predefined list to be executed.

6.1 List of Methods

For each parameter, a list of methods to analyze the parameter can be defined. For each method, the number of repetitions (number of measurements) can be specified.

Many relations between parameters and methods are possible.

One Parameter – Many Methods

For many parameters, multiple methods exist to execute the analyses. For example, to determine the moisture content (= parameter) of a sample, the analyst has the choice of:

- Gravimetric methods (dry & weigh)
- Near Infra Red (NIR)
- Karl Fisher
- …

The actual parameter result can then be calculated based on one or more method results.

Usually all methods are not always applied each time the parameter has to be analyzed. Often one quick and less-expensive method is applied to get an approximate indication about the parameter result. Only in case of an unexpected result, an additional method is added to obtain a result with higher porecision. For instance, when determining moisture in food samples NIR is applied as a fast method. In case of unexpected results, a method offering greater precision, such as Karl Fischer, is applied.

This can be configured in Unilab by a combination of assignment frequencies and alarm handling.

One Parameter – One Method

Sometimes a method is parameter-specific and cannot be applied to any other parameter. This is a simplified case of the previous situation.

One Method – Many Parameters

Some analysis methods yield results for multiple parameters. This is e.g. the case for the method chromatography. When running a chromatogram, the analysts can get data on more then one component (= parameters) in the sample.

One Parameter – No Method

Some parameters do not have a specific method. For example, this is the case for sensorial analysis such as taste and smell. The analyst just fills out the result (**OK** or **not OK**) on the parameter level.

Note

The method applied to execute the analysis sometimes depends on the sample type or the parameter profile. The **PP-PR-MT bridge** allows defining parameter profile specific methods. In case a method is defined on the **PP-PR-MT bridge**, the method list defined on parameter level is ignored.

6.2 Parameter Properties

6.2.1 Identification of the parameter

A parameter at the operational level has no meaning of its own and is therefore always referenced within its context. The complete identification of a parameter consists of the sample code, the parameter-group name, the parameter name and the node numbers at all hierarchical higher levels (pgnode and panode). The parameter node is used to distinguish two instances of the same parameter within the same parameter group. The node also determines the order (this eliminates the need for alphabetical sorting).

In addition to a name, a parameter gets a description, which is again copied from the corresponding parameter definition.

6.2.2 Processing Details of a Parameter

Processing details of the parameter are first of all provided by its current status in its life cycle. The default life cycle model used for newly-created parameters is set at the parameter definition level. Authorized users can always assign another life-cycle model to the parameter.

The date that the parameter was created and the responsible person for the parameter creation are also kept in the parameter properties. For parameters that have been assigned manually, the assignment information is included. The start of the analyses (entering results on parameter of method level) and the eventual completion of the analyses are also supplied automatically.

In case re-analyses have been performed for a parameter, the number of re-analyses is automatically filled out in the parameter properties.

6.2.3 Result Handling

Result Entry

The parameter result corresponds to the analysis result that will be reported. This result can be obtained in various ways. A result can either be entered at the parameter level, or calculated using values entered at the method level.

If results are entered at the method level, the calculation method used to obtain the actual parameter result based on the individual method results must be specified. This includes defining the number of measurements required to trigger the calculation and the actual calculation algorithm.

Number of measurements to trigger calculation

The number of measurements necessary to calculate the parameter result can be set in the parameter definition level. The minimum number of method results specifies how many method results need to be available before the parameter result is calculated (for the first time). In case the number of methods exceeds the minimum number of method results specified, the previously-calculated result is set aside as a re-analysis.

Note

The only exception to this rule is the calculation of an average. This calculation is triggered only as soon as all assigned methods have a result.

Example: In order to obtain the result of the sample parameter **FFA,** a titration is repeated three times. The parameter result must only be calculated after the third method result becomes available. The minimum number of results required for parameter-result calculation is thus set to **3**.

The number of repetitions of a method can depend on the sample type. The minimum number of results can therefore also be set to **-1**. Its value is then set automatically to the (initial) number of assigned methods for an individual sample.

Example: Measuring the **pH** of a water sample, will be done with a **pH_probe** (= method). This method is only repeated once. When measuring the **pH** of a dirt sample, an extraction precedes the actual analysis. The measurement is then repeated three times. For the parameter **pH**, the minimum number of method results to trigger the calculation is set to **–1**.

Important

The initialization of the correct minimum number of results is only done once. This means that adding extra methods to an existing sample parameter does NOT influence the minimum number of result values.

Calculation algorithm

Concerning the calculation algorithm, a parameter result can be calculated in several ways based on the available method results:

- First executed method or last executed method: the result of the first / last executed method is taken as the parameter result (after the number of methods specified in the minimum number of measurements field have been finished, i.e. have been executed or cancelled)
- Highest accuracy level: the result of the executed method with the highest accuracy level is taken as the parameter result (the accuracy level is defined as a property of the method itself)
- Average: the average of all the executed methods is calculated to get the parameter result
- Custom calculation: if none of the aforementioned options is applicable, a custom function can be defined to calculate the parameter result
- No calculation: if no calculation is required (in case the method result must be copied to the parameter result).

Custom calculation functions are PL/SQL functions stored at the database side. To calculate the parameter result, the float value of the method result is always used.

Formatting parameter results

Parameter results are stored in two different formats in the database: as a (text) string and as a (binary) number. An entered parameter result (string) is first converted to the appropriate float value and subsequently formatted.

For parameter results, long text results are supported as well.

Tip

Refer to the chapter on Formatting values for more details on the conversion process and the different formats.

Specification handling

An entered parameter result is evaluated against the set specifications. In case the specifications are violated, an alarm can be triggered. The alarms for each of the specifications are defined in the parameter definition.

The fact that the parameter result lies outside the specifications can also be used in conditions to trigger a number of state transitions in the parameter life cycle (e.g. to the status **Out of spec**).

Cancelling a parameter

Parameters that should not be measured can be cancelled. All corresponding methods are automatically cancelled as well. For the cancelled methods, the worklist assignment rules are then evaluated. For each of the cancelled objects, the execution end date is supplied. If the execution start date was still blank, this date is filled out as well.

Results of cancelled analyses are not used for charting and SQC.

Reanalyses of a parameter

For a parameter that has a result out of specifications (or for many other reasons) a re-analysis can be performed. The existing parameter result(s) are then copied to a separate table in the database. Old results from parameters that are reanalyzed are not used for charting and SQC.

All assigned methods are also re-analyzed: hence, the worklist assignment rules are re-evaluated.

The parameter status upon re-analysis depends on the life cycle definition applied. The method list on re-analysis can be specified through actions in the parameter life cycle.

6.2.4 Dynamic Assignment of Parameters

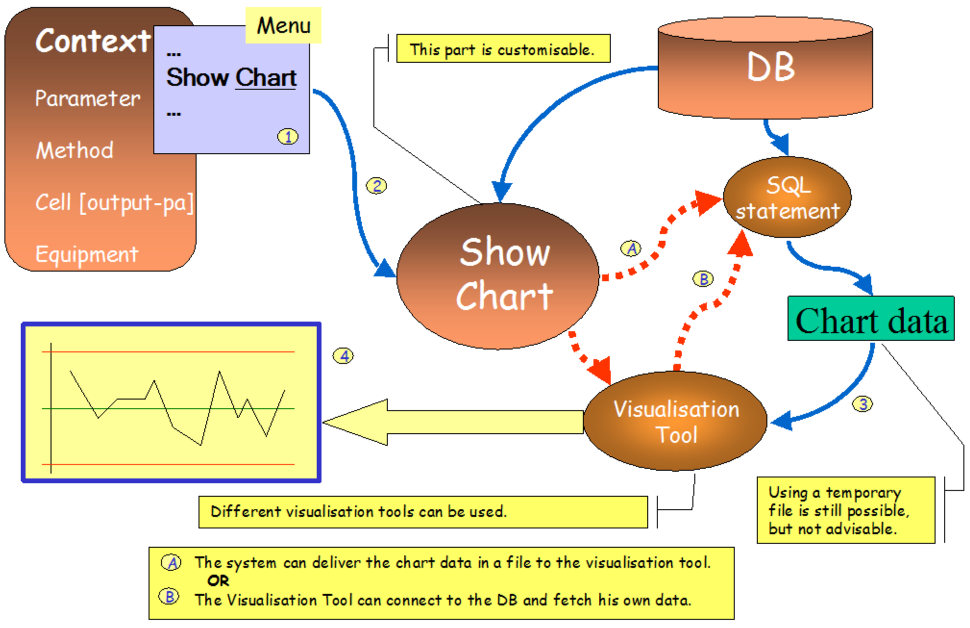
Parameters can be assigned dynamically to parameter groups at the operational level. The user should have the appropriate functional access rights to do so.

The list of parameters to choose from is controlled through the **Assign any parameter** flag on the parameter profile level.

6.2.5 Charting

Several charts can be called for parameter results at the operational level.

The picture below shows the mechanism of charting for a parameter.


Context
Parameter
Method
Cell [output-pa]
Equipment
Menu
...
Show Chart
...
1
2
This part is customisable.
DB
SQL statement
Show Chart
A
B
Chart data
Visualisation Tool
3
4
Different visualisation tools can be used.
Using a temporary file is still possible, but not advisable.
A The system can deliver the chart data in a file to the visualisation tool.
OR
B The Visualisation Tool can connect to the DB and fetch his own data.

When a chart is defined for a certain parameter, this chart can be displayed by selecting the chart option (1) in the right-click menu on the parameter level, for example, in the parameter result window. If multiple chart types are assigned to the parameter, the different series of Chart Types are displayed. The summary info of the last chart of the serie is fetched : chartid, visualization function, chart type, …

Selecting a serie retrieves the custom function ShowChart (2) that will decide which visualization must be executed. This will show the Chart dialog.

This dialog displays a graph showing the evolution of the parameter results for the selected parameter over a range of samples. You can scroll through a control chart by using the navigator on top of the Chart dialog. When the dialog displays a trend, the navigator is not active.

When displaying the chart dialog for parameterresults of a sample created in the past, you will always see the last chart of the serie. Note that “cancelled” parameter results are ignored, and thus not included in the Chart dialog.

For more information on charting, please see the **SIMATIC IT Unilab – The Charting Module** manual in the Unilab Product Library.

6.3 Parameter Definition Properties

6.3.1 Parameter Properties

The properties of the parameter itself conform to the generic object properties. Most of the properties that are set in the parameter definition refer to the operational parameter.

6.3.1.1 Result format and unit

The format and unit in which the parameter is measured can be set on the parameter definition level.

Tip

Refer to chapter on Formatting and Conversion of Results in the **SIMATIC IT Unilab -Concepts Guide Part 3** for more information.

6.3.1.2 Default Value

For some parameters, a default value is applied which can be set in the parameter definition. This default value can either be a fixed value or a value based on a user-defined attribute. The latter option is used if the default value is context dependent (e.g. dependent on the sample type). If a user-defined attribute is used to establish the default value, the user must specify the object type for which the attribute is defined and the name of the attribute.

Example

A company produces two products (= sample types) with a different package size. The package size is defined as a parameter. The default value of this parameter depends on the sample type. The default value is set as the value of a sample type attribute called **Package size**. The parameter definition refers to this attribute to establish the default value.

6.3.1.3 Confirmation of Manual Assignment

When parameters are manually assigned at the operational level, user confirmation might be required. This is controlled through a flag in the parameter definition: when the **User ID must be confirmed** flag is set, a user at the operational level will be prompted to enter his/her user ID and password to confirm the assignment. If the user ID is not identical to the logon ID, the parameter will not be assigned.

6.3.1.4 Assignment of Methods to a Parameter

The automatic assignment of a method to an operational parameter is controlled through assignment frequencies at the configuration level.

At the operational level, a method can be assigned ad hoc to a parameter. The user must have the appropriate functional access rights to do so.

The method selection list the analyst views at the operational level is fetched from the configuration level. In the parameter definition, it can be set whether this list should be restricted to the methods that have been explicitly assigned to the parameter definition or not. This is achieved by setting the **Allow any method** flag at parameter definition level. Note that in most cases, the restriction will be applied due to the fact that many methods are parameter-specific.

6.3.1.5 The checkbox Don’t assign if in other parameter

Activating the checkbox **Don’t assign if in other parameter** on pr–mt link results in the fact that once the method has been assigned to one parameter of the parameter group, it is no longer assigned to the remaining parameters. The method is assigned to the parameter in which it is created first, but not to the other parameters of the same parameter group.

The behaviour of the checkbox **Don’t assign if in other parameter** is based on 3 typical scenarios:

Ring Analysis

This scenario occurs when the same physical monster is tested in different labs.

Desired behaviour: the associated methods should not be assigned if another Pa already has this method within the same parameter profile.

Example

Below is an example of ring analysis.

```
Sc
     - pgX (in-house)
          pa
               - me
          …
     - pgY (external)
          - pa
               - me
```

Duplo Analysis

This scenarion occurs when some tests are executed more than once on the same physical monster.

Desired behaviour: if the multiple analysis returns multiple results, a new Pg will be used to keep them. When the analysis returns only 1 result, there is no problem.

Example

Below is an example of duplo analysis.

```
sc
     - pg1
          - pa (2X)
               - meX     //1 result = OK  - if multiple
results => save in other pg
               - meX
     - pg2 (if multiple results)
```

With fixed parameter list

This scenario occurs for some sample types for which a fixed list of test methods is defined. For example, in the case of metals, the list of test methods is ‘legally’ defined and hence may not be modified.

Desired behaviour: the same methods should be used in all Pg, without assigning them again.

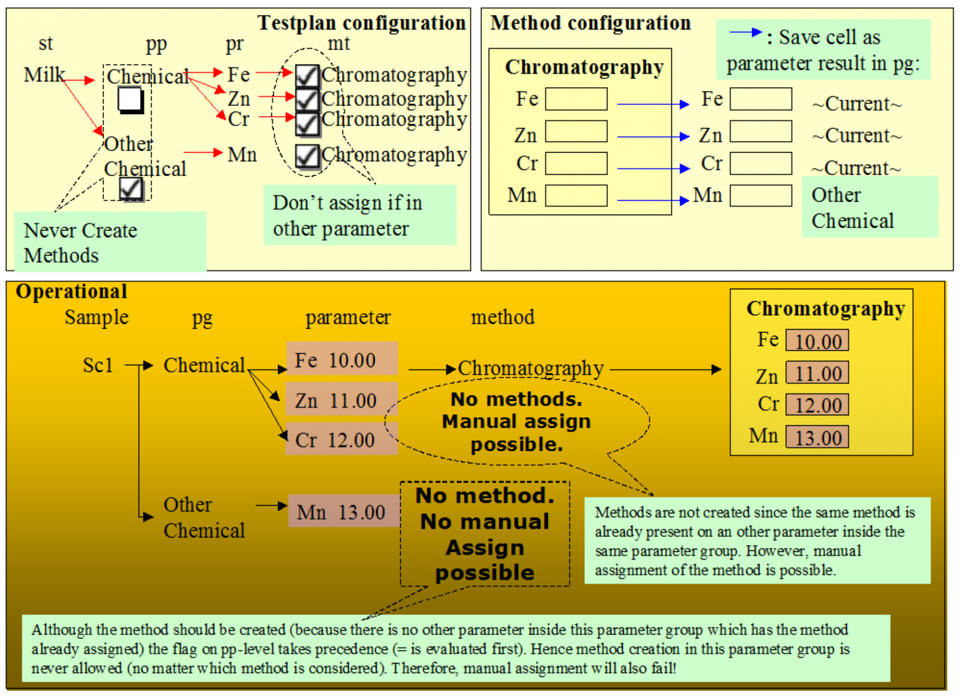
Example

Below is an example with a fixed parameter list.

```
sc
     - pg Metals   (list of parameters is ‘legally’ fixed, 1
method)
          - pa1
          - pa2
          - …
          - paN
               - pg Metals+additions
                         - pa N+1 // Never create methods
                         - …
                         - pa NN
```

Summary

The picture below show an overview of the behaviour of the checkbox **Don't assign if in other parameter**.



Methods are not created since the same method is already present on an other parameter inside the same parameter group. However, manual assignment of the method is possible.

Although the method should be created (because there is no other parameter inside this parameter group which has the method already assigned) the flag on pp-level takes precedence (= is evaluated first). Hence method creation in this parameter group is never allowed (no matter which method is considered). Therefore, manual assignment will also fail!

The checkbox **Don't assign if in other parameter** means that the method will not be created for this parameter if it is already assigned (and the status of the method is different from **Cancelled**) to another parameter for the same parameter group.

The checkbox **Never create methods** means that the parameter group may never contain a method.

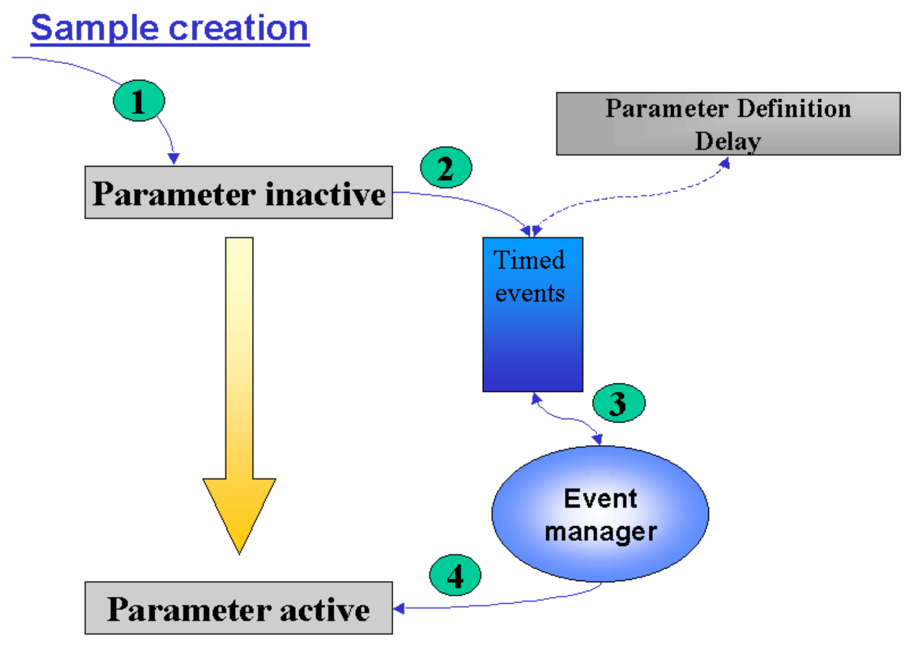
The multi-run functionality will be limited to the scope of one parameter group (not illustrated in this figure).

6.3.2 Delay Handling

Not all parameters should become active at sample creation. For example, analyses for shelf-life testing or stability studies are performed at regular time intervals (e.g. after one week, after two weeks, etc.).

For these parameters, a delay can be set. This delay specifies how long after the sample sampling date the parameter should become active.

The figure below illustrates the mechanism of delay handling on a parameter.



On sample creation, the parameter is created as well (1), but the initial state will be **"@D Delayed**" (even if this state does not occur in the life cycle). It is implicitly assumed that the **Delayed** status is not active. A timed event, specifying the parameter delay, is inserted into the timed event table. The event manager evaluates the entries in this table at a specified interval (3). Once the delay is over, an event is triggered that activates the parameter (through a state transition to the next/initial state (4)).

For methods assigned to a parameter that has a delay, the worklist assignment rules are evaluated as soon as the parameter becomes active. The methods themselves also become active (because their life cycle is also evaluated) and are assigned to the appropriate worklist(s).

Example

When establishing a bacteria growth curve, the microbiological growth is established at regular intervals (after 1 hour, after 2 hours, etc.). For each measurement, a new parameter is configured, called e.g. **Growth_1H**, **Growth_2H**, etc. For each of these parameters, an appropriate delay is set (1 hour, 2 hours etc). At sample creation, only the parameter **Growth_0H** is activated. All other parameters are activated once their time has elapsed.

Tip

A delay can also be specified at the parameter-group level. Note that delays at the parameter level and parameter-group level add up: a parameter delay is only counted from the moment that the parameter group becomes active.

6.4 Alarm Handling

Notifications (alarms) can be triggered when certain parameter results are out of specification. Unilab allows specifying how the system must react when an alarm situation arises:

- Message on the status bar
- Prompt to enter comment
- Warning message (that needs user confirmation)
- Trigger of re-analysis
- Custom alarm handling.

Custom function alarm handling permits defining customer-specific logic procedures, such as adding other parameters or parameters groups, logging on more samples or assigning a more accurate method to a parameter to confirm the result out of spec obtained with a less accurate method.

6.4.1 Alarm Definition

The alarms triggered when a parameter result is out of specification are defined in the corresponding parameter definition. For each of the specifications in the three specification sets, a different alarm can be set. The priority at which the specification sets are treated is also defined at the parameter level.

Note

The specifications themselves are not part of the parameter definition. Specifications are set for a parameter assigned to a parameter profile.

The alarm themselves are custom functions. Both VC++ and PL/SQL custom functions are supported for alarm handling.

Tip

For more info on developing custom alarm handling functions, please refer to manual **Customizing the System** in the Unilab Product Library.

6.4.2 Standard Alarm-Handling Mechanism

Each time a parameter result is entered, it is checked against the specifications for that parameter. In case the result is out of spec, the alarm-handling mechanism is triggered.

The specifications have been exceeded and the direction in which they have been exceeded is kept by means of nine flags on the database side (one flag for each of the three specifications in each of the three spec sets).

Example

When the low limit of spec set A has been exceeded, the value of **Valid_limitsA** flag is set to “**<**” for the corresponding parameter.

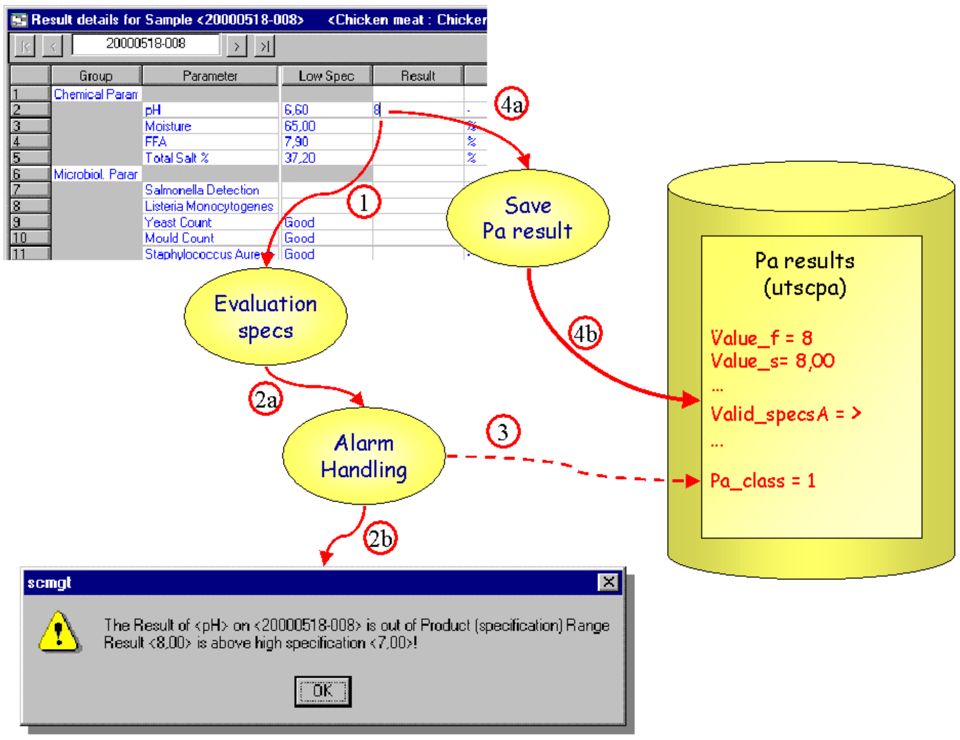
The fact that the alarm has been handled is set by means of the **pa_class** on the database side. The value for this field is set to **1** in case the alarm has been handled for a specific parameter.

The actual order of events for alarm handling depends on whether the result has been entered directly on the parameter level by the end user, or whether it is obtained in another way. The latter case includes situations such as parameter results that are provided through an instrument connection or results that are obtained through calculation based on method results.

Result entered on parameter level

The mechanism described here is applied in case the parameter result is filled out on the parameter level by an end user in the Unilab applications.

The figure below illustrates the mechanism for alarm handling.


Result details for Sample <20000518-008> <Chicken meat : Chicker

20000518-008

	Group	Parameter	Low Spec	Result	
1	Chemical Parar				
2		pH	6,60	8	-
3		Moisture	65,00		%
4		FFA	7,90		%
5		Total Salt %	37,20		%
6	Microbiol. Parar				
7		Salmonella Detection			
8		Listeria Monocytogenes			
9		Yeast Count	Good		
10		Mould Count	Good		
11		Staphylococcus Aure	Good		-

scmgt

The Result of <pH> on <20000518-008> is out of Product (specification) Range
Result <8,00> is above high specification <7,00>!

The following sequence of events occurs upon result entry:

When a result is entered, the specifications are checked. If the result falls outside the specifications, the system internally sets the correct values for the appropriate alarm-handling flags.

The appropriate alarm-handling function is called and executed.

The **Pa_class** property is set to **1** for the parameter.

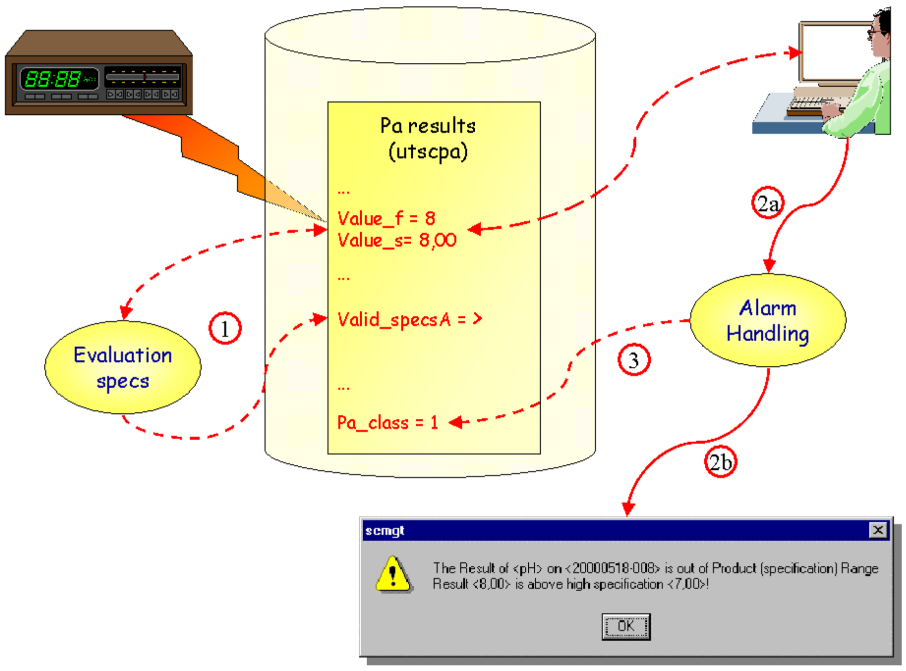
When the parameter result is saved later, also the settings for the alarm-handling flags are updated in the database.

Result provided by non-Unilab user

A parameter result can be obtained indirectly. For example, this is the case for:

- Parameter results obtained through calculation based on method results
- Parameter results provided by a remote application or instrument through **Uniconnect**.

The picture below shows an overview on how alarms are handled when the parameter result is provided through **Uniconnect**.



The Result of <pH> on <20000518-008> is out of Product (specification) Range
Result <8,00> is above high specification <7,00>!

In this case, the following sequence of events is triggered:

When the result is obtained, the specifications are evaluated. If certain specifications are exceeded, the corresponding alarm handling flags are updated in the database. Since the alarm function itself is not yet running, the **Pa_class** setting remains unmodified.

The first time the parameter result is accessed through the GUI (for instance, the first time a user opens the **Parameter results window**), the alarm is triggered and the necessary actions are executed. The **Pa_class** setting is then updated.

Alarm functions versus life-cycle actions

Special events that are triggered when a parameter result is out of spec can also be triggered by an action in the parameter life cycle. However, using life-cycle actions presents two major drawbacks compared to Unilab alarms:

- A life-cycle action is only triggered after the parameter result has already been saved to the database
- A life-cycle action does not evaluate the alarm-handling flags.

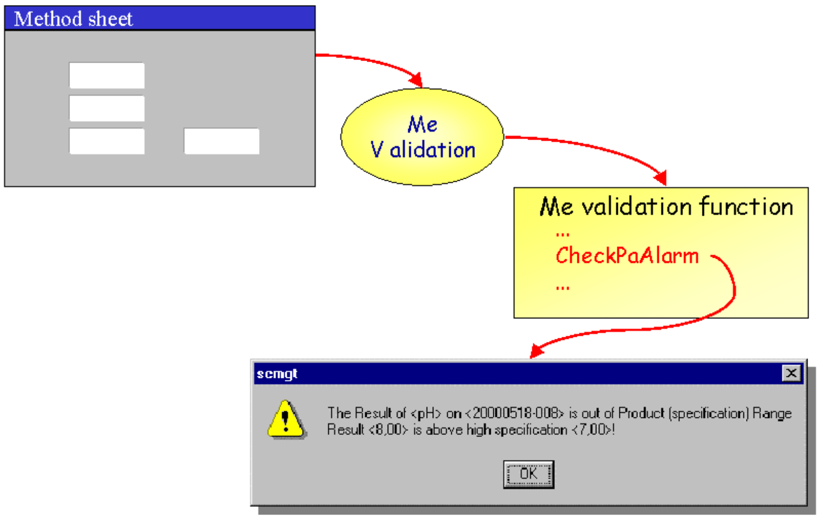
6.4.3 Checking Specifications on Method Level

By default, alarms are only triggered for parameter results. In some cases, user also wants to be notified about a result that is out of spec when:

- Filling in a method result used to calculate the parameter result
- Filling in the value for the method cell that will be saved as method result (for a method of that parameter).

This can be obtained by calling the **CheckPaAlarm** function from within a validation function of the method or method cell, respectively.

The figure below illustrates how parameter alarm handling is triggered upon saving a method result.



The Result of <pH> on <20000518-008> is out of Product (specification) Range
Result <8,00> is above high specification <7,00>!

6.4.4 Exception Logging

Unilab permits the logging of exceptions. Results out of specification are stored in a separate table. This provides a flexible base and good performance for the generation of exception reports.

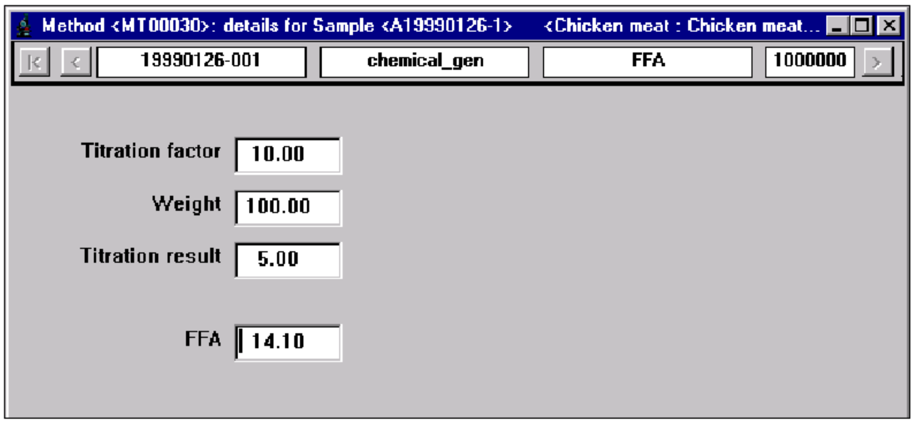
7 Methods

7.1 Introduction

A method defines how, step by step, a parameter result is determined. It usually consists of some (physical, chemical) measurements. Several raw data values are often used/combined in calculations in order to obtain the final (parameter) result. For one parameter, multiple methods can be defined. On the other hand, one method can also be used to determine different parameters.

The step-by-step execution details of a method are implemented through a method sheet. A method sheet is a free-format form containing several cells for data input and/or result calculation.

The figure below shows the method sheet for the titration method (MT00030) used to determine the Free Fatty Acid content (FFA) of a sample.


Method <MT00030>: details for Sample <A19990126-1> <Chicken meat : Chicken meat...

19990126-001	chemical_gen	FFA	1000000

Titration factor	10.00
Weight	100.00
Titration result	5.00
FFA	14.10

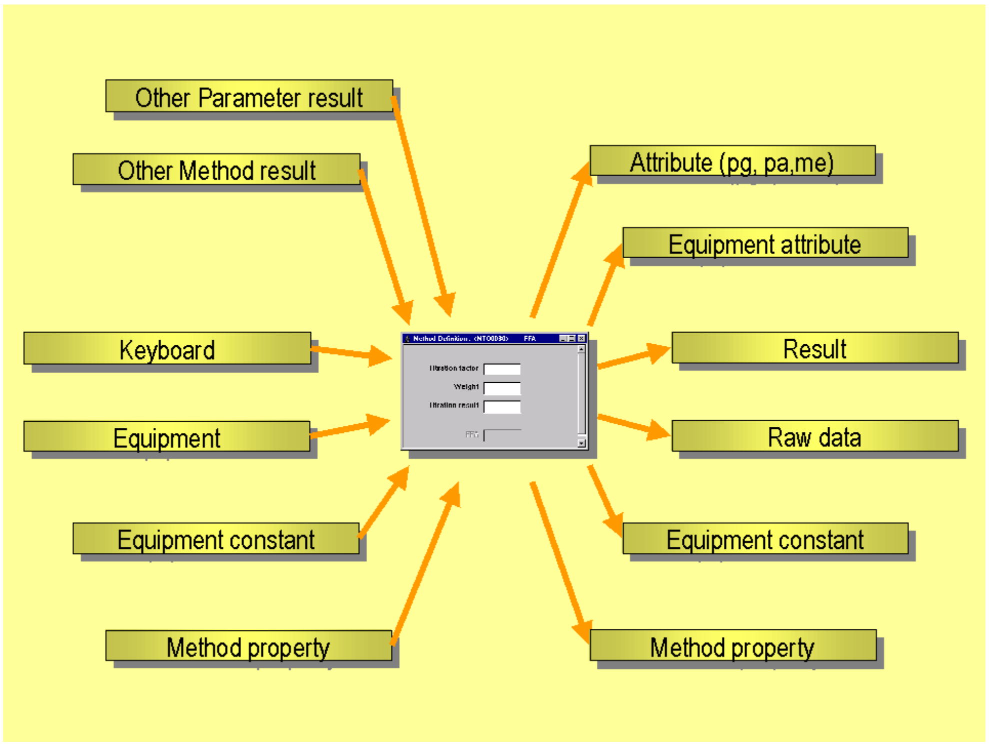
For each of the individual method cells and for the method sheet, a number of properties can be defined.

7.1.1 Method Input and Output

Because a method sheet implements the different steps of analysis execution, it may require input values from different sources. These input values are used to calculate preliminary or final results. The obtained results can, in turn, be distributed to other objects (e.g. method sheets) for further processing.

What source a method sheet gets input from or to what target a method sheet provides output for is handled on the method-cell level.

The picture below shows the input and output options for cells in a method sheet.



Method input using the keyboard

The most common input is provided through the keyboard. Even if another source is specified, it always remains possible to supply the value by using the keyboard (unless the cell is protected).

Other analyses results as input

Complex method sheets, in which calculations are performed based on previously-obtained parameter or method results, require these values as input. For example, some methods require the weight or pH of a sample in order to calculate a correct result.

Equipment

In modern laboratories, many analyses are performed by means of an analytical instrument of a specific equipment type. The values obtained from these instruments are then used as input for method sheets where they are further processed.

Equipment constants

For method sheets where an equipment of an equipment type provides input values, certain equipment type specific constants must be taken into account to calculate the actual value. These constants can be the coefficients of a calibration curve or the correction factors on that instrument. These values are implemented as equipment constants in Unilab and can be used as input for method cells.

The equipment constants themselves are obtained through the execution of specific interventions on the instrument. These interventions can be implemented by specific intervention methods. An example is a calibration method used to establish the slope and the intersection of a calibration curve. The information/ results obtained from a calibration method must be stored for the equipment it was executed on. So, the obtained or calculated values in the corresponding cells can be saved as output for the equipment constants themselves.

Saving method-cell values as result

The main reason an analytical method is executed is to provide an analysis result. The value contained in the method cells can therefore be saved as a result.

Most methods provide a result for the parameter to which they are assigned. There are however also methods that provide a result to multiple parameters. An example is chromatography where an analyst can obtain the content of multiple components (parameters) in one sample in one run. The values contained in each cell will then be saved each as result of another parameter.

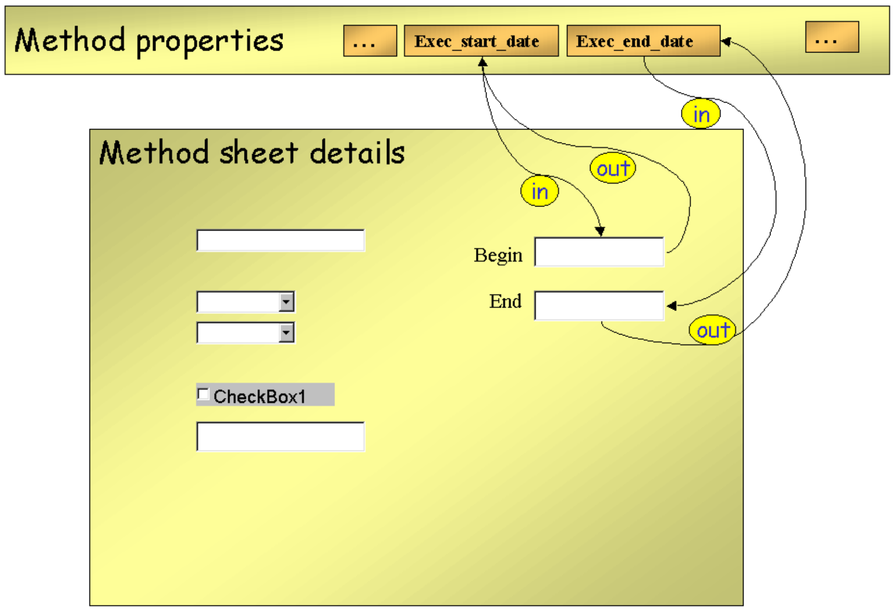
Attribute values as method sheet output

Methods can also yield values that need to be stored as additional (user) attributes (instead of results). The individual method-cell values are then saved as attribute values for the appropriate object attributes. The objects are limited to parameter groups, parameters and methods.

Linking method cells to method sheet properties

It is sometimes necessary to modify method properties together with entering the sheet details. To ease the entry of method properties in combination with cell values, it is possible to link specific cells with method properties.

The picture below shows how to link specific method cells to method properties.



Examples are a cell showing the executor name or the execution start and end dates of the method.

Raw data

The method-sheet details (= the method-cell contents) are stored in special database tables. These details include pick lists, cell properties and method-cell values used for calculation. The tables in which the details are stored are dynamic tables used for fast and efficient processing of operational method sheets. Due to the complexity of these method-sheet details, they should not be used/accessed for reporting purposes. Other reasons that discourage using these details are:

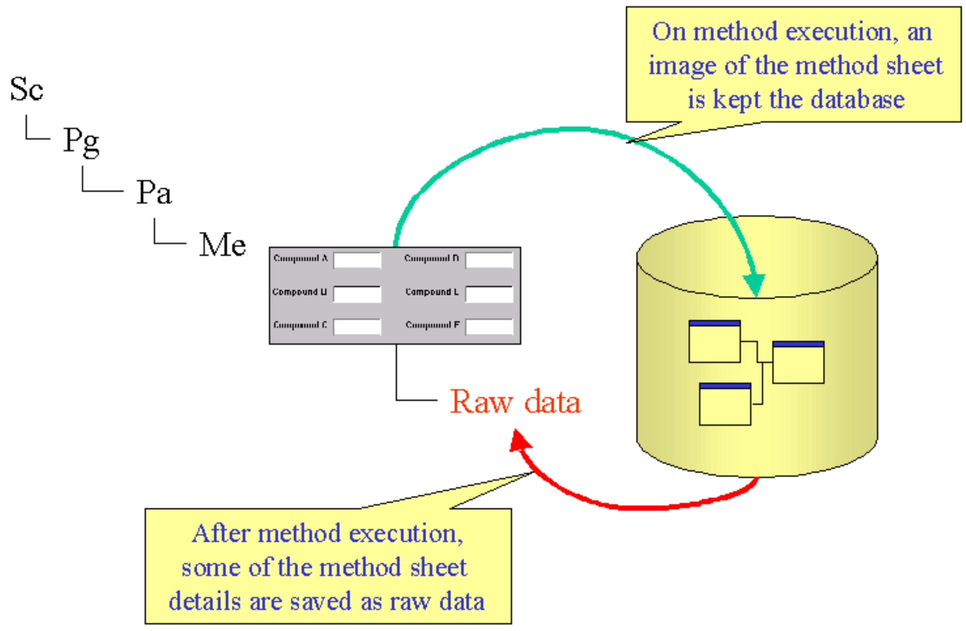
- The details can be removed from the database.
- There is a lot of overhead information that is purely used for visualization purposes.
- The structures are likely to change when the system is upgraded to a new version

Note

There is no automatic procedure that deletes the method-sheet details after method execution. If this is required, it must be handled by a special action in the method life cycle or during the archiving procedure of operational data.

Specific cell details can be stored as “raw data” which are easy to use/include in queries. This option can be selectively set per method cell.

The figure below illustrates the concept of raw data of a method sheet.

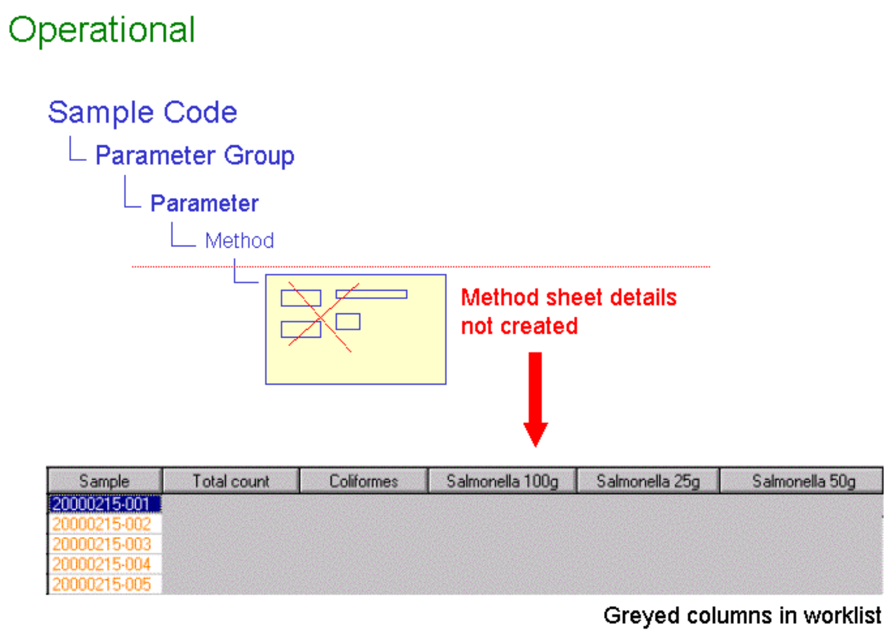

On method execution, an image of the method sheet is kept the database
Sc
Pg
Pa
Me
Compound A
Compound D
Compound B
Compound E
Compound C
Compound F
Raw data
After method execution, some of the method sheet details are saved as raw data

7.1.2 Method Sheet Creation

At the operational level, methods are created during parameter creation according to the configuration level assignment frequencies. For performance reasons, the method details (method cells and associated properties) by default are not created at method creation. Method details are only created when the method sheet is opened for the first time. This “creation on demand” better distributes the average load on the system.

However, in some circumstances, the method details should be created at method creation. For example, this is the case when method-list layouts are used that display the method cells. The purpose of such layouts is to provide a fast and efficient way of supplying method-cell values without having to open the corresponding method sheets. However, without special measures, the columns corresponding to the method cells in the method-list layout will be greyed (i.e. disabled) for newly-created samples as long as the method sheet has not been opened.

The figure below shows the greyed columns in a worklist due to the absence of method details.



Sample	Total count	Coliformes	Salmonella 100g	Salmonella 25g	Salmonella 50g
20000215-001					
20000215-002					
20000215-003					
20000215-004					
20000215-005					

However, ithe creation of the method details can be forced by setting the **Create cells automatically** standard property in the method sheet.

7.1.3 Group-key Assignment and Worklist Assignment

Group-key assignment

Methods at the operational level have group keys. This enables the analyst to select a specific list of methods in a worklist. These group keys are not inherited from the configuration level, as is the case for samples, for example. Upon method creation, a number of method group-key assignment rules are evaluated. The group-key assignment rules define which group keys with which group-key values should be assigned to newly-created methods.

Example: To be able to select a list of methods for a specific workday, a method group-key assignment rule is defined. This assignment rule assigns the group key **Workday** to newly-created methods. As a value, the first six characters of the sample code (**YYMMDD-XXX**) are copied.

Worklist assignment rules

A group-key assignment rule is evaluated only once, more specifically when the method is created in the database. However, the worklist concept demands a dynamic assignment to and removal of methods from the worklists. Whether a method should be present on a worklist depends on the status of the method (e.g. when a method is created, it is automatically assigned to an equipment worklist; as soon as the result is available, the method is removed from this worklist again). Therefore, special assignment rules are defined based on the method life cycle.

Worklist assignment is implemented by assigning values to method group keys. These rules are evaluated whenever a state transition occurs in the method life cycle.

Example: A method should be present in the worklist of the analyst marked as planned executor in the method properties when it has the status **Created** and it should be removed from the worklist when it has the status **Executed** or **Cancelled**. The following assignment rule is configured:

Method Status	In	Out	Group key	Group-key value
Created	X		Analyst	Planned executor
In Execution	X		Analyst	Planned Executor
Executed		X	Analyst	
Cancelled		X	Analyst	

7.2 Method Properties

7.2.1 Identification

A method at the operational level has no meaning on its own and is therefore always referenced within its context. The complete identification of a method consists of the sample code, the parameter-group name, the parameter name, the method name and the corresponding node numbers at all levels (pgnode, panode and menode). The method node is used to distinguish two instances of the same method assigned to the same parameter. In addition to a name, a method gets a description, which is copied from the corresponding method definition.

7.2.2 Properties of a Method

The properties of a method are filled out in two ways: some properties are filled out upon method creation, other are filled out during method execution. Users with the appropriate access rights can update most properties at any time, provided that the method is still modifiable (**allow_modify flag** = 1).

Properties supplied at method creation

Many method properties are automatically supplied at method creation. For example, this is the case for audit information, such as the assignment date, the person responsible for assigning the method and the flag indicating whether the method has been assigned manually or automatically. Other properties are inherited from the corresponding method definition upon method creation. For instance, this is the case for:

- The standard operating procedure (SOP), describing how the method should be executed
- The plausibility limits
- The formatting information (format and conversion factor)
- The information for worklist assignment (planned equipment and planned executor)
- The accuracy level of the method.

At the operational level, users having appropriate access rights can modify each of these properties, with the exception of the SOP. For example, the laboratory manager can modify the worklist assignment information in order to assign the method to another worklist.

Properties filled out during method execution

A number of method properties are automatically supplied during method execution. This includes the execution start date, the execution end date and the actual method executor. The actual equipment used to obtain the method result is only supplied when equipment has been specified in the method definition property sheet.

Users having the appropriate access rights can enter some information themselves (e.g. the actual time and cost of method execution). By comparing these values with the estimated time and cost set at method definition level, a more accurate workload planning can be established.

7.2.3 Automatic versus Dynamic Assignment

At parameter creation, methods are automatically assigned to the operational parameter as specified by the configuration level assignment frequencies.

Methods can be dynamically assigned to operational level parameters. The user should have the appropriate functional access rights to do so. The list of methods to choose from is controlled by the **Assign any method** flag on the parameter definition level. As the majority of the methods are parameter-specific, this flag is not set in most cases. This avoids assigning an inappropriate method to a parameter.

7.2.4 Method Execution

A number of checks are triggered during method execution. These checks are listed in the paragraphs below in their order of execution.

Authorization check

Whenever an attempt is made to modify the property sheet (this includes not only all standard properties of a method, but also the properties of each individual cell), the system will first verify whether the current user is authorized to execute the method. If the current user is not authorized (duce to lack of experience or qualifications), the user will be warned that he/she is not authorized to execute the method and the modification will be reversed/undone.

If the authorization list is empty, any user is allowed to enter results and modify the method in any way. Note that the actual executor is always filled out.

Equipment check

When the **Equipment** (= actual equipment field) of the method is still blank, the system will check if there is any equipment specified for this method. If there is no equipment specified, the **Equipment** field is automatically filled with **"-"** to indicate that there is no equipment specified. This avoids further repetitive evaluations.

If more than one equipment has been specified, the user will be invited to specifically select one of the listed equipment.

Check of equipment status

If an equipment has been selected to execute a method, then the actual operational status of that equipment will be checked. The following situations may occur:

- The equipment is active. In this case, the method can be executed.
- The method is not a calibration method AND the equipment is not active. In this case, the method cannot be executed until the equipment becomes operational (active) again.
- The method is not a calibration method AND the equipment is active, but the warning level of the equipment differs from zero. In this case, the method execution can proceed, but the user is informed about the actual situation. The user is prompted to perform the required equipment intervention first
- The method is a calibration method. The status of the equipment is ignored because, most likely, this method is executed just to get the equipment "calibrated" again.

7.2.5 Result Handling

Result entry

The results of a method can be obtained in two ways: immediately at the method level or in the method sheet (by supplying all cell details and calculating the end result). The latter option is usually applied for complex methods, where the actual method result is obtained through calculations performed on one or more input values.

When the result is entered at the method level, the **manually entered** flag in the method properties is set.

Formatting

Method results and method-cell values are stored in two different formats in the database: as a (text) string and as a (binary) number. An entered value (string) is first converted to the appropriate float value and subsequently formatted. For method results, long text results are supported as well.

Tip

Refer to the chapter on Formatting values for more details on the conversion process and the different formats.

If the method cell **Component** property is not empty, the value entered in the method cell will be checked against the measurement range for the specified component and appropriately formatted.

Cancelling methods

Methods that should not be executed or aborted (for any reason) can be cancelled. When a method is cancelled, the worklist assignment rules are evaluated. The effect is that cancelled methods are removed automatically from the worklists. For each cancelled method, the execution end date is supplied. In case that the execution start date was still empty, this date is filled out as well.

Results of cancelled methods are not used for parameter result calculations.

Reanalysis of a method

The execution of a method can be re-started at any moment, even when the final result is already available, by performing a re-analysis. The existing method result is then copied to a separate table in the database. These old method results are not used for parameter calculations. For the reanalyzed method, the worklist assignment rules are re-evaluated and the method is automatically re-inserted into the correct worklists.

The status the method gets on re-analysis depends on the life-cycle definition applied. The transition from the current status to this “re-analysis” status can have authorization restrictions imposed. If this is the case, only authorized people can perform a re-analysis.

Complete methods

The cells where an output attribute is specified are only copied to their final output destination when the method is “completed”. The default and automatic definition of “completion” is that all output cells have a value assigned. Output cells are all method cells for which the value is saved as method result, parameter result, equipment constant…

Tips

A table cell is considered as completed when at least one cell has been filled out (contains a value).

Hidden cells are ignored in the evaluation of the completion.

The **exec_end_date** of a method is filled out automatically when the method sheet is marked completed. An actual method result is not required.

The **Confirm completion** flag overrules the default behaviour, prompting the user (when the method sheet details are saved) to confirm completion. Upon answering **Yes**, the method will be saved with the **Completed** flag set (manually).

Complete methods: special rules for method sheets belonging to a worksheet

By default, when a method belongs to a modifiable worksheet, and the **Confirm completion** flag is not set, then no cell output distribution is performed when all output cells are filled. Such distribution will take place only when the status of the worksheet changes to a non-modifiable status (e.g. approved).

However, in some specific cases, the worksheet must be ignored in order for the methods to be completed. This is supported by standard Unilab, but requires setting the column **utws.ws_class** to 1 or 2 (through a simple update in a trigger or any other way).

With **ws_class** = 1, the methods can be completed before the worksheet is completed, but any method or worksheet event will evaluate whether the methods making up the worksheet must be completed or not.

With **ws_class** = 2, the methods can also be completed before the worksheet is completed. But, in this case, the distribution of events to all the other methods in the worksheet will not take place.

The latter possibility (**ws_class** = 2) solves the performance issue in the case of large worksheets (worksheets with **ws_class** = 1 and containing e.g.: 100 samples with 20 methods), where method completion is required, but there is no need to distribute this method result to the other methods in the worksheet.

Note The **Confirm completion** flag of the method overrules the **ws_class** setting.

Calculation in method sheets

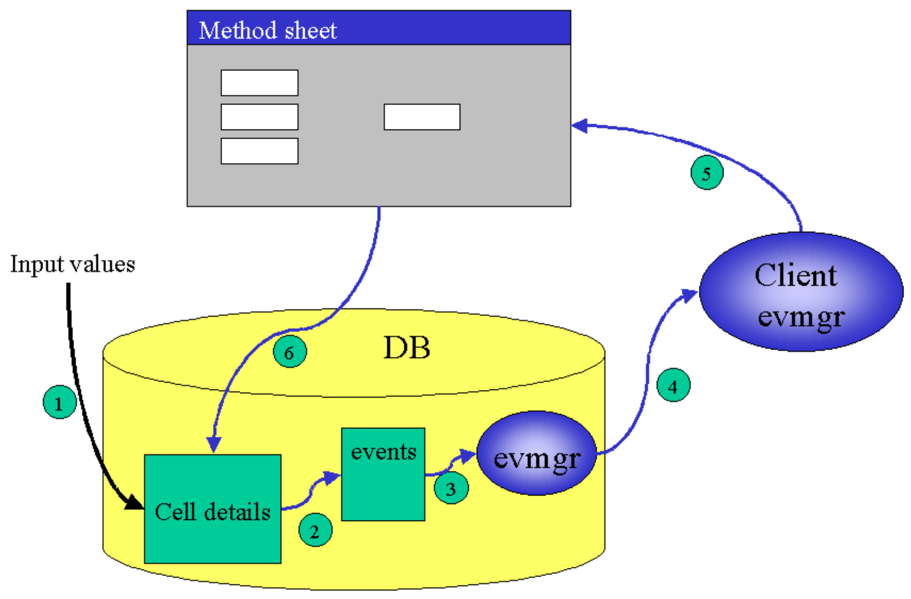
Some cells in the method sheet contain calculated values (using other cell values). The calculation can be a simple formula defined in the formula interpreter or a more complex algorithm (containing procedural logic).

The **Calculate automatically** flag indicates whether the calculation of the calculated field is performed each time any cell value changes or not. By default, the calculation is triggered automatically.

Calculations (whatever type) are immediately performed whenever any method-cell value changes. The calculations are performed according to the tab order of the cells. This is important if a calculated value is a preliminary result used for further calculation in the same method sheet (since changing the order of the calculation will result in a different result).

In case method-cell values are automatically acquired (using the Unilink module or any other type of custom interface), the system also provides a way to automatically update the calculated fields. These calculations are executed by the client event manager.

The picture below shows the method-sheet calculation.



Whenever method-cell values are inserted/updated in the database and the method sheet is not completed, the client event manager will automatically load the method-sheet details into (its own local) memory and automatically re-calculate intermediate results. Any modifications (due to the calculations) will automatically be saved to the database. Note that when automatic completion has been specified, the DB-API which saves these calculated results will automatically handle (and distribute) the cell outputs (if any had been defined).

Note that calculations are always performed when a method is just loaded into memory and the **automatic calculation** flag is set!

7.3 Method Definition

The properties of the method sheet itself conform to the generic object properties.

7.3.1 Standard Properties of a Method

Worklist assignment settings

Some standard method properties are used for worklist assignment at the operational level. Both the planned executor and the default equipment can be set for assignment to the correct worklist. The system will verify that the default equipment occurs in the specified list. If necessary, a warning will be issued and the planned default equipment will be appended to the list.

Properties for workload management

In the method properties, an indication can be given about the estimated cost and estimated time for the execution of the method. This information is used for preparing worklist estimates.

Standard operating procedure

In a certified laboratory, procedures for method execution are saved as ISO9000 or GLP documents. These documents can be made available on line for consultation during method execution. The link to these documents is provided through the appropriate service attribute (**unmedoc**).

Calibration methods

Quality regulations state that analysis should only be performed on analytical instruments that are regularly calibrated and/or for which a scheduled maintenance or intervention plan has been established.

In many laboratories, well-established procedures exist to execute these equipment interventions. These procedures can be implemented in Unilab using a method.

Any defined method can be used for equipment calibration and intervention. However, Unilab is conceived in such a manner that no methods can be executed on equipment that is inactive. Equipment becomes inactive when a scheduled intervention has not been executed satisfactorily.

The purpose of methods implementing interventions is to make the equipment active again. To differentiate these methods from analytical methods, which should not be performed using inactive equipment, these methods can be defined as calibration methods. Methods explicitly set as a calibration method are the only methods that can be executed on an equipment that is not active.

Result format and unit

A display format and a measurement unit can be specified for each method result. The unit on the parameter level can be different from the unit defined for the individual methods. A conversion factor to convert the method result to the parameter result can be defined.

Tip

Refer to chapter on Formatting and Conversion of Results in **SIMATIC IT Unilab - Concepts Guide Part 3** for more information.

For individual method cells, default formats can also be specified.

Plausibility limits

Physical constraints apply to many analyses. For example, a pH result should always have a value between 0 and 14; a percentage can never be higher than 100. Plausibility limits can be defined in the method properties.

The plausibility limits are used at the operational level to verify the entered result. They play an important role in the elimination of typing errors. The analyst is notified when the entered value exceeds these boundaries and the entered value is not accepted by the system.

Default value

For some methods, a default value is useful because often the same (default) result will be obtained.

The default value can be set as a fixed value, entered in the method properties. In many cases, however, the default value is context-dependent. For example, it depends on the sample type or on the parameter. A default value that depends on the context in which the method is used can be set using a user-definable attribute.

This attribute is assigned with the appropriate value to the specified object. In the method properties, reference is made to the attribute and the object type to which it is assigned.

At the method-cell level, only fixed default values are supported. Note that only single value cells can have a default value: a table type cell cannot have a default value.

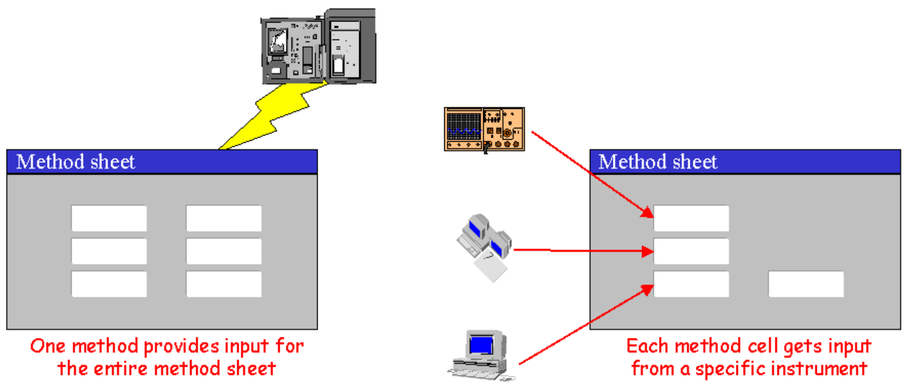
7.3.2 Equipment for Method Execution

For each method, a set of instrument types can be specified to choose from on the moment of method execution. On method execution, the right equipment is selected based on the specified equipment type and the current lab context.

Equipment setup

There are basically two ways to set-up a link between a method and analytical equipment.

The picture below shows the strategies for equipment setup.


Method sheet
One method provides input for the entire method sheet
Method sheet
Each method cell gets input from a specific instrument

A number of analytical methods require a set of equivalent instruments that can be used to perform the method. This implicitly assumes that the entire method only requires input from one single instrument. This is, for instance, the case for methods using chromatographs. All cells in the method sheet get their value from one and the same instrument. On the operational level, however, the user can choose from several equivalent chromatographs to actually execute the method.

Another category of methods requires a combination of instruments in order to obtain all required raw data. This can be handled by specifying, for each individual method cell, the equipment from which it will get its input. If several different instruments are used to obtain different parts of raw data for a method, the instrument list can be specified at the level of the individual cells. An example is a titration method, in which the weight of a sample is obtained using a balance, and the titration value is obtained by means of a titrator.

In situations where the method requires input from several instruments (in whatever form), the planned equipment (at the method level) could still function as the “principle” instrument that can be used for worklist assignment. If any information about the specific setup of the instrument is required, this information must be maintained in separate cells of the method sheet.

Logical versus electronic connection

Note that an analytical instrument does not need to be connected electronically in order to be able to trace where the results came from. Even if the results are entered manually, it remains possible to establish the proper (logical) link to the equipment from which the data originated.

Tip

The equipment and equipment-type definition itself, including the communication protocol with the equipment, is set and managed in the **Define equipment** application.

Measurement ranges

Each analytical instrument has measurement ranges for tdetermining a specific component. However, the applied analytical method has a substantial influence on these measurement ranges. Therefore, Unilab offers the possibility of defining measurement ranges at the method level. When the system detects any measurement-range specification for a specific component at the method level, the measurement-range specifications (for that component) on the equipment are overruled.

Tip

Although the measurement ranges are set at the method level, they only apply to the individual method-cell values. For each method cell, it is possible to specify the component for which the measurement ranges should be checked.

Equipment selection at the operational level

At the operational level, a user can select the equipment type for method execution. This establishes the logical link between an analysis result and the equipment (type) used to generate that result.

To avoid the user selecting the equipment type for each individual input, the following rules are applied:

- When there is no equipment type configured, the equipment type field will be automatically filled with **“-“.**
- When only one piece of equipment type is configured, the system will automatically select that equipment (only after evaluating the operational status/availability of the equipment)
- When input can be supplied by more than one piece of equipment type (according to the current configuration), and the **current equipment** occurs in this list, the system will always (automatically) assume that this new measurement also comes from the same equipment (as the previous one).

Example: In this manner, a user can move through a worklist and first determine the initial weight for a set of samples. Next the user can move to another equipment (by selecting it just once) and then perform the second step in the method (for the same set of samples).

7.3.3 Authorization

The first check during method execution at the operational level concerns authorization. Unilab verifies whether the current user is trained to execute the method and, hence, to modify the method properties. On the method-definition property sheet, it is possible to specify who is authorized to execute the method.

The authorization to execute the method can be specified as the level of experience that is required to execute the method. However, it is possible that a user has not – yet – acquired a specific experience level, but that he/she is already allowed to execute a specific method. Therefor, the authorization can also be granted to specific users or to user profiles. It is also possible to make any number of combinations of the above options.

Any user who does not have the required experience level and who is not explicitly authorized to execute the method, will not be allowed:

- To make any modification to the method properties
- To change the cell contents.

Tip

When the authorization list remains empty, all users are authorized to execute the method.

7.4 Method Cells

7.4.1 Types of Method Cells

Several method-cell types are supported in Unilab. For each of these method cells, a number of properties can be defined.

Tip

For the definition and use of the different types of method cells, please refer to the On Line Help of the Configuration application.

7.4.2 Standard Attributes of Method Cell

The standard attributes include the name of the cell, the description (title) and a second description. The second description is only available for reporting purposes.

Tip

The same method-cell name can be used in different method sheets (within one and the same method sheet, the name must be unique). However, keep in mind that method cells with the same name must be of the same type if they are used in layouts (for instance, worklist layouts).

The width and the height of the method cell in the method sheet can be set in the standard attributes tab. Note that cells in a method sheet can also be sized and positioned using standard Microsoft Windows functionality.

For each method cell, a default value can be specified. This default value is always a fixed value, entered in the method cell itself at the configuration level.

A cell can also be flagged as protected and/or hidden. These properties can be modified at the operational level. However, they can be toggled dynamically within custom calculation functions.

A method cell can also be marked as mandatory (similar to the info fields). Method-sheet details cannot be saved when mandatory info is missing. The check for mandatory information is only implemented on the client side; the DB-APIs do not take the mandatory flag into account.

7.4.3 Component Specification

For each method cell, a property **Component** can be set. The component is used to establish the link with the measurement ranges (if any have been specified).

It would be a logical naming convention for the matching component to use the name of the cell. However, sometimes it is more convenient to use a different name. Therefore, the list of currently-existing components is available as a pick list.

To allow the user to specify the component name of a cell before having specified the measurement ranges, the **Component** field can be freely edited.

The special value **~current_pa** will be dynamically replaced (during execution) with the actual parameter name (to which this method is assigned). This permits dynamic checking against different component values, depending on the parameter that is being determined.

Whenever a cell value is modified, the system will automatically verify whether or not the result falls within the specified measurement ranges. For method cells, the client application will always (first) apply the normal conversion and formatting rules. If the **Component** field of the cell is not empty, a number of additional checks will be applied automatically (before the custom validation function is called).

If the measurement ranges for the specified component are defined at the method level, these will be used to format the result accordingly. Otherwise ,the system will try to fetch the measurement ranges from the equipment corresponding with this cell. In case there is no equipment for the current cell, the equipment at the method level (if any) will be used as an alternative.

Tip

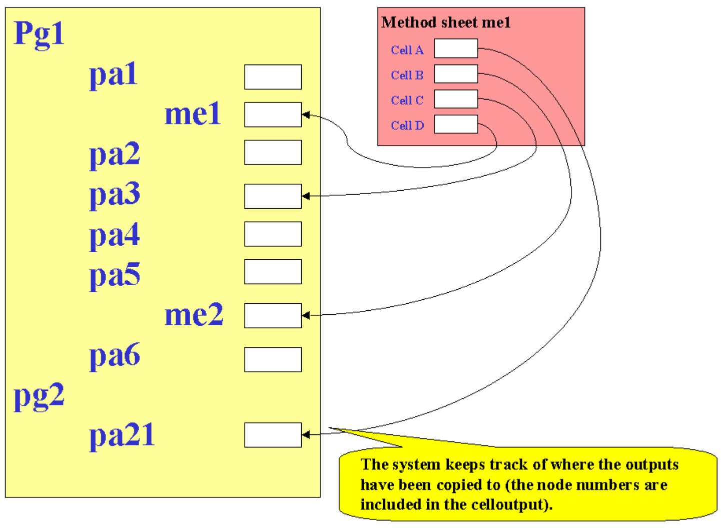
The actual formatting rules when measurement ranges have been specified, are discussed in the chapter on Formatting and Conversion of Results in **SIMATIC IT Unilab - Concepts Guide Part 3**.

7.4.4 Method-Cell Handling

Method-cell input and output

Each method cell can get input from various sources and can, in turn, provide output to a target. Unilab keeps fully-detailed tracing for each cell input and output. This detailed tracing includes the name of the input source /output destination and all node numbers of the involved hierarchical levels (pgnode, panode and menode).

The figure below illustrates method-cell handling.


Pg1
pa1
me1
pa2
pa3
pa4
pa5
me2
pa6
pg2
pa21
Method sheet me1
Cell A
Cell B
Cell C
Cell D
The system keeps track of where the outputs have been copied to (the node numbers are included in the celloutput).

Input attributes

Each method cell can get input from one source. If the specified input is missing, the method cell remains blank. Method-cell input can come from the following sources:

- The keyboard
- A parameter result
- A method result
- A method property
- An equipment constant
- Equipment

In case input is provided by another object in the Unilab database, the input source is identified through qualifiers. These qualifiers can be set for each method cell individually.

Output attributes

Each method cell can, in turn, provide output to one target. The value of each method-sheet cell can be saved in a specific/different location in the database. By default, the cell value is not saved. The output source for a method-cell value is determined using qualifiers.

In case the method-cell value is saved as result, the qualifiers determine the combination (parameter group/ parameter/ method) of where a cell is saved. The most obvious option is to save a method-cell value as an analysis result. The method-cell value can be saved as the final result of the method sheet. Qualifiers allow the user to select the parameter profile, the parameter and/or the method for which the result should be stored. By default, the qualifiers are set to **~Current~**,indicating that the parameter group, parameter and method to which the method sheet has been assigned will be used at the operational level.

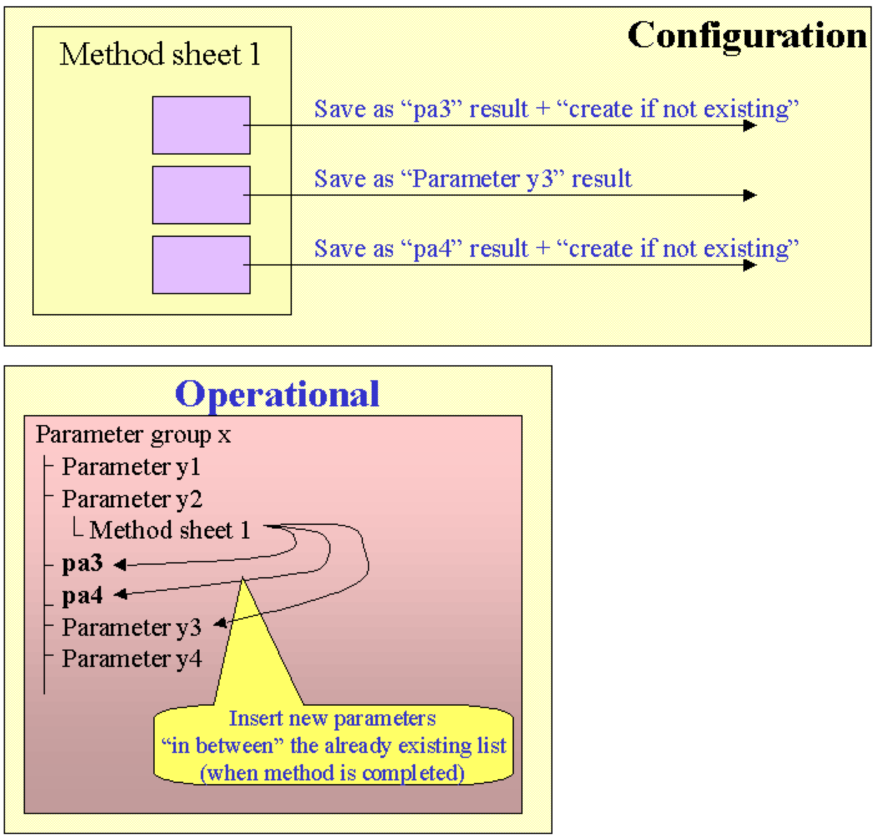
Dynamic creation of cell output

In case the cell contents are saved as parameter or method result, it is possible that the corresponding object might not yet exist in the database. For example, this is the case for qualitative analysis methods. These methods are used to determine what compounds (=parameters) are present in a certain sample. Hence, the method can yield an analysis result for a parameter that has not yet been assigned to the sample.

If the output target/record is not present, it can be created dynamically. On the method-cell level, the option **Create new entry** allows the dynamic creation of objects set as output target in the method-cell properties. This option can be set for the following levels individually:

- Parameter-group level
- Parameter level
- Method level

The figure below illustrates cell-output handling.


Configuration
Method sheet 1
Save as "pa3" result + "create if not existing"
Save as "Parameter y3" result
Save as "pa4" result + "create if not existing"
Operational
Parameter group x
Parameter y1
Parameter y2
Method sheet 1
pa3
pa4
Parameter y3
Parameter y4
Insert new parameters "in between" the already existing list (when method is completed)

Tip

If the **Create new entry** check box is NOT selected and the parameter profile/parameter/method combination does not exist, Unilab will NOT create the corresponding objects.

If this check box is selected and the parameter profile/ parameter/method already exists, Unilab will not create the corresponding object also in this case.

Tip

For table cells, the object (parameter, method,…) will only be created for objects that actually have a result (rows containing a value. E.g.: the following table cell is an output cell for parameter results.

```
+++++++++++++++++++++++++
+C    1      Ppm        + => parameter is created
+NO2  40     Std units  + => parameter is created
+NO3         Pbb        + => parameter is  not created
+SO4  1935   Mg/l       + => parameter is created
+PO4         ppm/cp     + => parameter is not created
+++++++++++++++++++++++++
```

Reanalysis/Cancel versus method output

The full traceability supports propagation of re-analysis and cancel transactions. When a parameter **pa1** or method **me1** is reanalyzed, the parameters/methods set as output qualifiers for the method cells of **me1** are reanalyzed as well (automatically). Of course, the corresponding life cycles (of the target parameters and methods) must allow for a re-analysis/cancel transition starting from their current state.

7.4.5 Result Attributes

Result attributes include the value of a cell (float and string) and the display format. Method cells can accommodate for long text results. A validation function runs on client side. It is triggered as soon as the corresponding method cell loses the focus.

7.4.6 Calculation Formula

For calculated fields, the formula used to obtain the cell contents must be defined. The calculation formula can only be based upon the input values of other cells in the same method sheet.

Simple calculations can be edited using the formula interpreter. For more complex calculations, a calculation function can be defined. This is either an VB script or a VC++ custom function run on the client side.

7.4.7 Tabbing Order

The number of cells in a method sheet is unlimited. The order in which the cursor moves from one field to the next when **[Tab]** is pressed can be modified.

The tab order is very important for a method sheet. First of all, the setting of a good tab order increases the efficiency with which the data can be entered. However, it also determines the order in which calculations are performed. This is very important in method sheets in which a number of preliminary results are calculated. The preliminary results will be calculated in the order in which the corresponding fields get the focus.

7.4.8 Method-Cell Audit Trail

The method-cell audit trail is part of the method audit trail, as this is more user-friendly and considering that method cells are properties of methods.

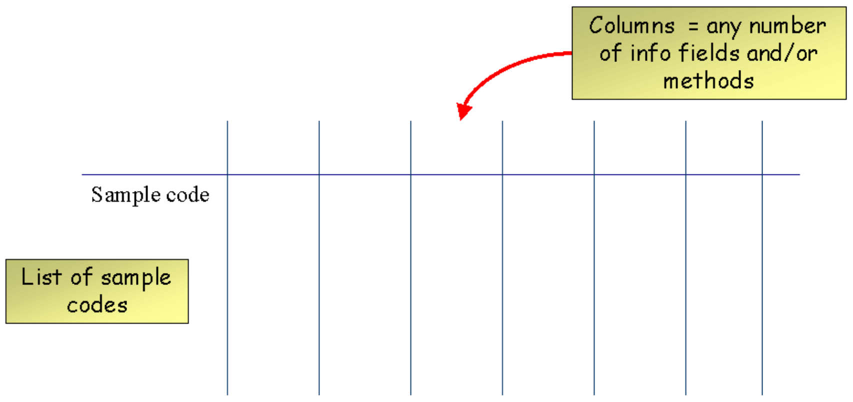
8 Worksheets

8.1 Worksheets and Worksheet Types

8.1.1 Worksheet

In essence, a worksheet is a sample list that permits displaying samples, sample group keys, info fields, methods, method cells and method group keys, all in one layout. It is a cross table with sample data, info fields and methods.

The figure below shows the structure of a worksheet.



A worksheet is not only used to present a set of data, but also to directly enter the results on a cell-by-cell basis. For simple methods, it is possible to enter the result of the methods here manually or even to activate the instrument. For others, the method sheet can be opened. Also the sample info card and parameter result window are directly accessible from within a worksheet.

In order to maintain a fast and easy accessible link between all samples in a worksheet, each worksheet is identified by a unique code number. Moreover, each worksheet has a life cycle of its own. This life cycle keeps track of the overall status of sample processing and analysis execution for that worksheet.

8.1.2 Worksheet Type

A worksheet type is a template definition containing all the properties of a worksheet of that specific type. It allows simplifying and speeding up the process of creating new worksheets.

8.1.3 Worksheet versus Worklist

Both worksheets and worklists serve as data-entry screens for analysis results. Nevertheless, each object has different properties and, thus, different applications.

A worklist displays the workload for an analyst, for an instrument, for a work day, etc. Each entry in the worklist (each method) stands on its own. Each time the state of a method in the list changes, a set of worklist assignment rules are evaluated. If required, the method is removed from the worklist. Removing a method from a worklist has no influence on the overall properties of the worklist itself.

A worksheet groups together multiple samples for which a set of similar analyses (methods) must be executed. The samples included in a worksheet are processed in a grouped way. The worksheet has an ID and a life cycle of its own. The worksheet life cycle depends on the status of the worksheet details (worksheet samples and worksheet methods). A validation function allows the entire worksheet to be validated as a whole.

Moreover, samples are always displayed in the worksheet they are assigned to, irrespective of their status. Since info fields can also be included, a worksheet is more suitable for presenting a data set than a worklist.

8.1.4 Application of Worksheets

Worksheets are among other things used for handling:

- Microbiological matrices
- Dilutions
- Analysis runs (e.g. on an instrument).

A microbiological matrix lists all of the microbiological analyses that must be executed on a set of samples. Each row of the worksheet contains the relevant info fields and the required microbiological analysis for one sample.

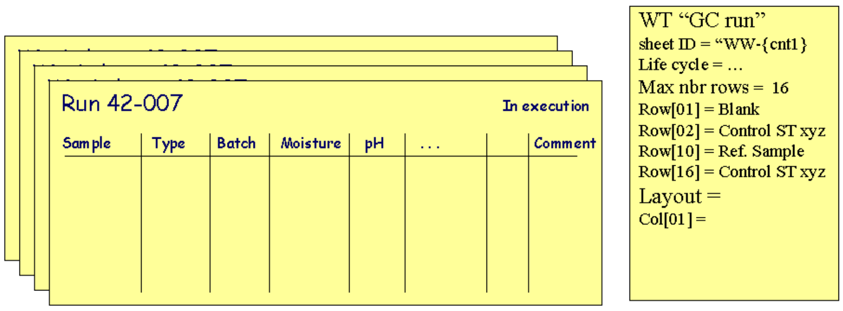
Sometimes a sample must be diluted in order to quantify the presence of a specific bacteria species. This situation only occurs when the standard method does not yield any result, or when the result is absolutely positive, but not quantifiable. In these circumstances, some experience is required to estimate at which dilution one will obtain a quantifiable value. Therefore, the range of dilutions is usually maintained. The dilutions are presented as columns in a worksheet for a set of samples. Note that the calculations required for dilutions must be implemented through custom functions.

Automatic analysis methods often permit the analysis of multiple samples in one run. An autosampler attached to a chromatograph, for instance, allows analyzing up to 50 samples during one run. These runs often contain one or more control and/or calibration samples to verify the validity of the analysis results obtained during the run. All samples contained in one run and the analysis details for those samples are displayed in a worksheet.

8.2 Worksheet Properties

A worksheet is essentially a list of samples combined with a definition of a layout (to determine the contents of the columns).

The figure below shows the properties of a worksheet.



At the moment of worksheet creation, all necessary samples are created and/or assigned to the worksheet. The database structure of an operational worksheet consists of one main table, listing the available worksheets and their properties, and three additional tables that hold the references to the different samples, info fields and methods contained in the worksheet.

The data contained in these tables define the data content of a specific worksheet. The worksheet layout determines how this data will be visualized in the application. This layout is set as a worksheet property.

A work sheet has a life cycle, access rights, a history and group keys. A worksheet has no info fields. Standard attributes and user-defined attributes can be used to store information on individual worksheets.

8.2.1 Identification of Worksheets

A unique code number is used to identify each worksheet. The format and algorithm used to generate the unique code number can be specified in a worksheet code mask.

Besides the unique worksheet code, each worksheet receives a description that is copied from the corresponding worksheet type.

8.2.2 Worksheet Layout

The columns and their properties in the worksheet layout are set through a layout definition. A worksheet-details layout determines how the data content of a worksheet is visualized on the screen. This layout is set as a property of the worksheet type.

The user can define the contents, the column title and the width of each column in the worksheet layout. A column can contain the following types of information:

- A sample property
- The value of a sample group key
- A specific info-field value (the info card cannot be specified)
- A method result (the parameter or parameter group cannot be specified)

- A method-cell value
- The value of a method group key.

By defining different layouts for a worksheet, the visualization of the worksheet details can be changed. However, the data content of these and other layouts must always be a subset of data referenced in the worksheet tables. When changing the layout of an operational worksheet, the data-content is not actualized to this new layout. If the layout contains fields that require data that is not part of the worksheet's data content, these fields will appear gray on screen.

Important

Also refreshing the worksheet will restore the initial layout of the worksheet.

8.2.3 Samples, Methods and Info Fields in a Worksheet

Identification of objects in worksheet

All objects in a worksheet are completely identified including their node numbers. This makes it possible to handle name duplication properly.

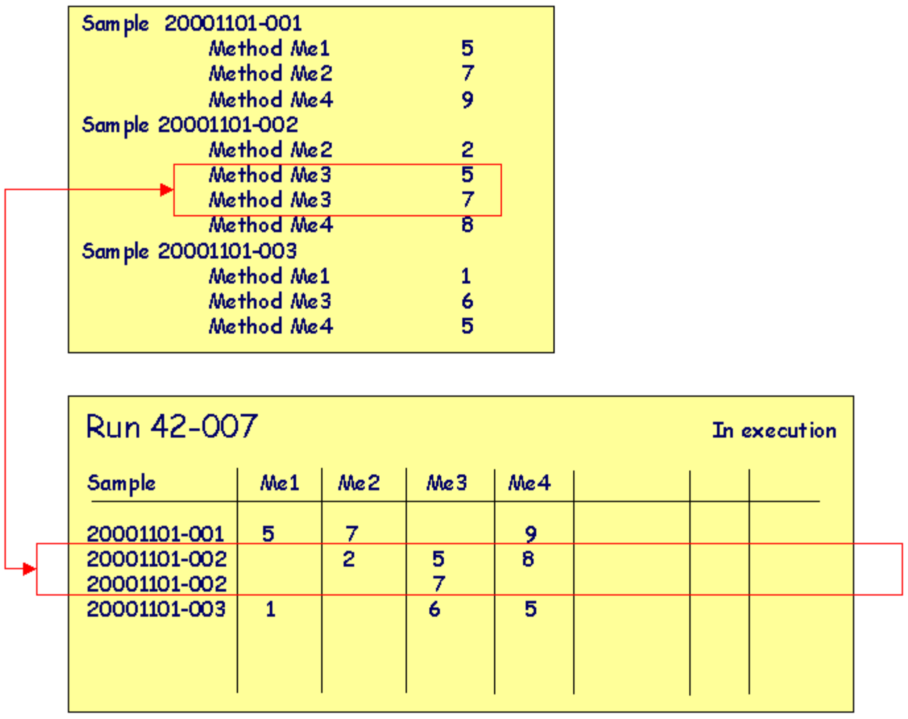
Access rights for objects in worksheet

When accessing a worksheet, the access rights on samples, methods and info fields are ignored. If someone is allowed to access a worksheet, he/she must have view access to all data contained in that sheet. This prevents erroneous conclusions or decisions from being made on the basis of partially-displayed data. However, when a worksheet is saved, the normal authorization checks for samples, info fields and methods are applicable. If, for example, the sample is not modifiable or if a user does not have **Write** access to a method, the modification made to the worksheet is not saved.

Duplicate methods in worksheet

The worksheet allows for duplicate methods. The duplicate method will be visualized in the worksheet.

The figure below shows a duplicate method in worksheet.



Sample 20001101-001		
	Method Me1	5
	Method Me2	7
	Method Me4	9
Sample 20001101-002		
	Method Me2	2
	Method Me3	5
	Method Me3	7
	Method Me4	8
Sample 20001101-003		
	Method Me1	1
	Method Me3	6
	Method Me4	5

Run 42-007 In execution

Sample	Me1	Me2	Me3	Me4			
20001101-001	5	7		9			
20001101-002		2	5	8			
20001101-002			7				
20001101-003	1		6	5			

Me3 of sample **20001101-002** is a duplicate method. Duplicate methods are visualized in the worksheet as different rows. Notice that all single methods of sample **20001101-002** are always visualized in the worksheet on the first row for sample **20001101-002**.

Maximum number of rows

For each worksheet, the maximum number of rows (samples) allowed in the worksheet (**max_rows**) can be specified on the worksheet type level. The **sc_count** field (sample count field) keeps track of the actual number of samples assigned to the worksheet (this is the number of rows already used).

For active worksheets, more samples can be assigned as long as the sample count is lower than the maximum number of rows. A worksheet is automatically marked **Complete** when the number of samples assigned to the worksheet reaches the maximum number of samples allowed (**sc_count** is equal to **max_rows**).

8.2.4 Worksheet Group Keys

Operational worksheets have group keys assigned. This enables the end user to select a specific list of worksheets.

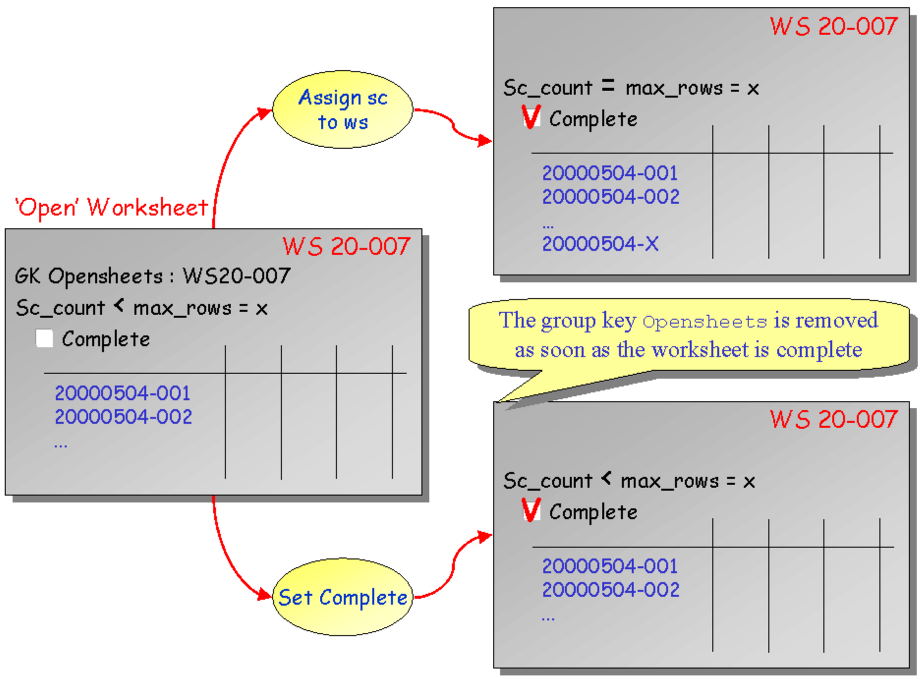
Group-key assignment

Worksheet group keys are not inherited from the configuration level, as is the case for samples. At worksheet creation, a number of worksheet group-key assignment rules are evaluated. The group-key assignment rules define which group keys (and which group-key values) are assigned to newly-created worksheets.

Opensheets group key

One worksheet group key, the **Opensheets** group key, is automatically maintained by the system to quickly retrieve the list of open worksheets on row assignment. Whenever a worksheet is marked **Complete** (the sample count of a worksheet reaches the maximum number of rows, or the worksheet is explicitly marked **Complete**), the corresponding worksheet entry is automatically removed from the group key.

The picture below shows the assignment of the **Opensheet** group key to 'open' worksheets.



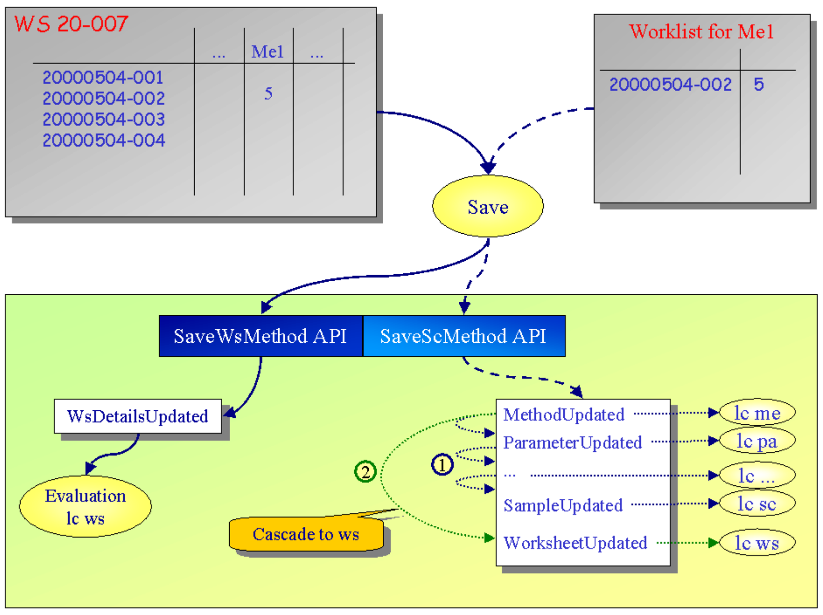
8.2.5 Worksheet Life Cycle

A worksheet has a life cycle of its own. State transitions in the worksheet life cycle are triggered by state changes in the life cycle of the worksheet details (samples and method assigned to the worksheet).

Events triggered on the worksheet level

When saving a worksheet, the method results included in the worksheet are saved into the database. The corresponding API (**SaveWsMethod**) triggers the necessary events to evaluate the worksheet life cycle. The events corresponding to **SaveScMethod** will be generated as well. This implies that the life cycles of the methods included in the worksheet are evaluated as well.

The figure below shows the worksheet life cycle.



Events triggered on the method level

Events triggered on the method level itself are propagated to the worksheet level. When a method result is, for instance, filled out in a worklist, the event is propagated to the worksheet level and the worksheet life cycle is evaluated.

However, the propagation to the worksheet level only occurs in a separate cascade operation after the event has already been propagated to the sample level (and, thus, after the sample life cycle has been evaluated). In this manner, the evaluation of the worksheet life cycle can be made dependent on the sample to which the method was assigned.

For example, consider a worksheet for FIA (Flow Injection Analyses) used to measure anions in a batch of samples. Each run is preceded by a blank and a control sample. Each run is terminated by a control sample. The worksheet is only closed (transition to status **Closed** in worksheet life cycle) once the last control sample reaches status **Validated**.

8.2.6 Worksheet Validation Function

A custom function can be defined to validate the worksheet as a whole. It is called after each modification to the worksheet. The worksheet validation function is a VC++ custom function.

An example of a worksheet validation function is a custom function, implementing the rules for the evaluation of microbiological dilutions.

8.3 Worksheet Type

A worksheet type is the template definition for specific types of worksheets used to speed up and simplify the creation of new worksheets. In addition, certain settings are provided to control the properties of the worksheet type itself.

8.3.1 Sample List for a Worksheet

A worksheet is essentially a list of samples combined with a definition of a layout (to determine the contents of the columns).

On the worksheet-type level, it is defined how the sample list for the worksheet will be established. This is done by either assigning sample types to the worksheet type, or by assigning well-defined, already-created samples to specific rows in the worksheet type. Control samples that should be included in every worksheet of a specific type can directly be assigned on the worksheet-type level.

8.3.2 Worksheet-Type Properties

The properties of the worksheet type itself conform to the generic object properties. The worksheet-type property sheet includes a number of settings that define the properties and the behaviour of the worksheets at the operational level. Each of these properties is set manually at the configuration level.

These settings include:

- The default life-cycle model for samples of this type
- The worksheet code mask used to generate the worksheet ID's
- The standard operating procedure describing how to handle worksheets of this type
- The worksheet layout applied for the worksheet details
- The minimum and maximum number of rows allowed in a worksheet of this type
- The custom function for validating the worksheet contents.

8.4 Worksheet Creation and Execution

A new worksheet is created based on a worksheet type. The worksheet type holds the default life cycle model used for the worksheet, the layout applied to visualize the data content of the worksheet and the unique code mask used to generate the worksheet ID.

Upon worksheet creation, all samples as specified in the worksheet type are assigned to the newly-created worksheet. The assigned samples can be existing samples, which have been explicitly assigned to the worksheet type, or newly-created samples based on sample types assigned to specific worksheet-type rows.

In practice, only generic samples (such as control samples in a run) are created and assigned automatically upon worksheet creation. All other rows will be filled manually upon worksheet execution.

8.4.1 Row Assignment

Rows (method and/or samples) can be assigned manually to a worksheet in different ways:

- Methods can be assigned from within a worksheet or a worksheet list
- Samples can be assigned from within the sample list application
- Samples/methods can be assigned from within a worklist.

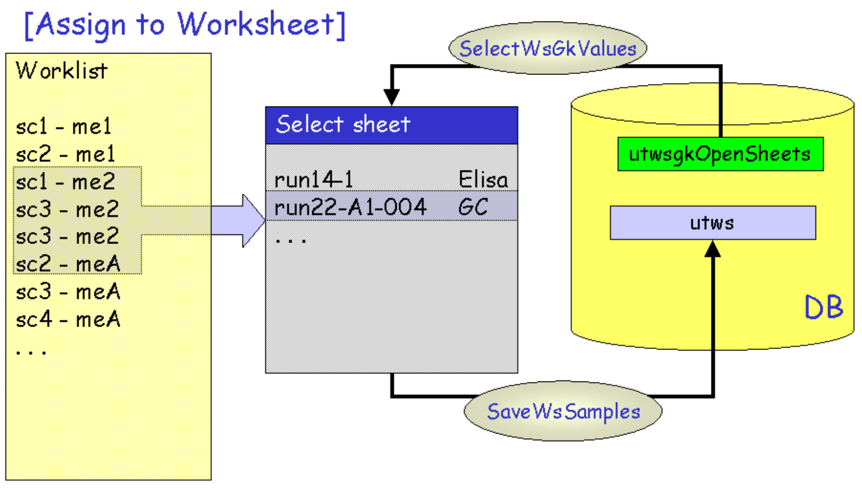
Rows can only be assigned to open worksheets (worksheets that are not yet complete).

Open worksheets / Complete worksheets

Because each worksheet has a specific page size (=maximum number of rows), samples can be assigned only to worksheets in which the actual number of samples **sc_count** is not yet equal to the page size. Worksheets that are not yet marked **Complete** are called open worksheets.

A special worksheet group key, **Opensheets**, is maintained by the system to quickly retrieve the list of open worksheets. The **Opensheets** worksheet group key is automatically assigned to newly-created worksheets that are not yet complete. It is automatically maintained by the system: each time additional samples are added to the worksheet, the assignment of the **Opensheets** groupkey is re-evaluated. Whenever the sample count of a worksheet reaches the maximum number of rows, the **Opensheets** group key is automatically removed from the worksheet.

The figure below shows how to assign samples to a worksheet.



However, sometimes it is necessary that an incomplete worksheet be locked from further addition of rows/samples. Therefore, a complete flag is maintained in the worksheet. When this flag is set, the **Opensheets** group key is removed from the worksheet, regardless of the value of **sc_count**. The complete flag can be used as a toggle so that the decision may be reconsidered afterwards.

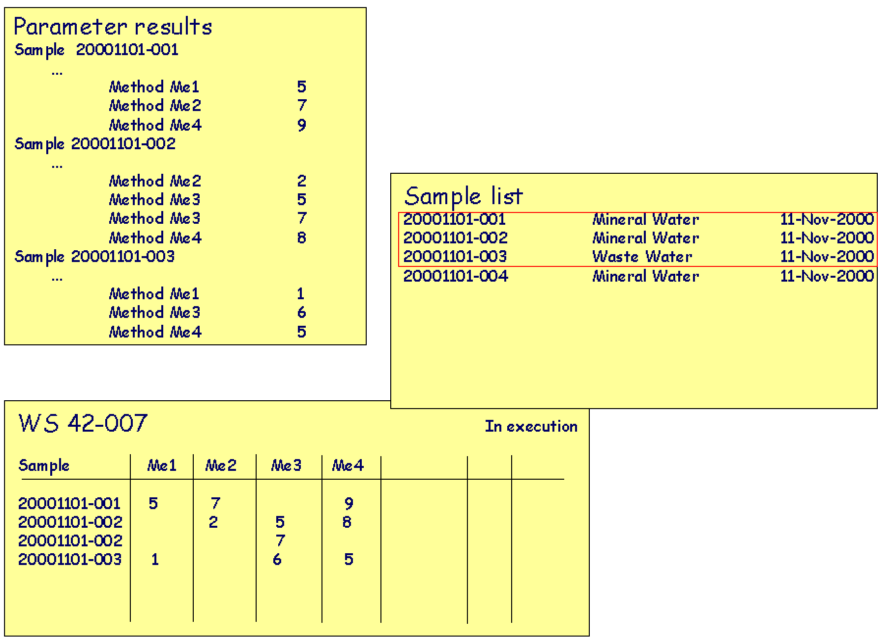
Row assignment

Samples or methods can be assigned to a worksheet from within a worksheet or worksheetlists. The sample list /method list displayed depends on the setting of the **wsAssingScTask** / **wsAssignMeTask** preferences on the worksheet task level.

Each worksheet task can have one of the following preferences:

- **wsAssignScTask = <Sample List Task>**: in this case, the group keys of the **Sample List Task** (hidden on **Sample list** window) and the matching **Sample List Layout** will be used to present the user a sample list for assignment of rows to the worksheet. As illustrated in the example below, the selection of 3 rows can result in an assignment of 4 rows in worksheet (duplicate methods) in the condition that the duplicate method (**Me3**) is part of the worksheet layout

The picture below shows how to assign samples to a worksheet from within a worksheet.



Parameter results

Sample 20001101-001
...

Method Me1	5
Method Me2	7
Method Me4	9

Sample 20001101-002
...

Method Me2	2
Method Me3	5
Method Me3	7
Method Me4	8

Sample 20001101-003
...

Method Me1	1
Method Me3	6
Method Me4	5

Sample list

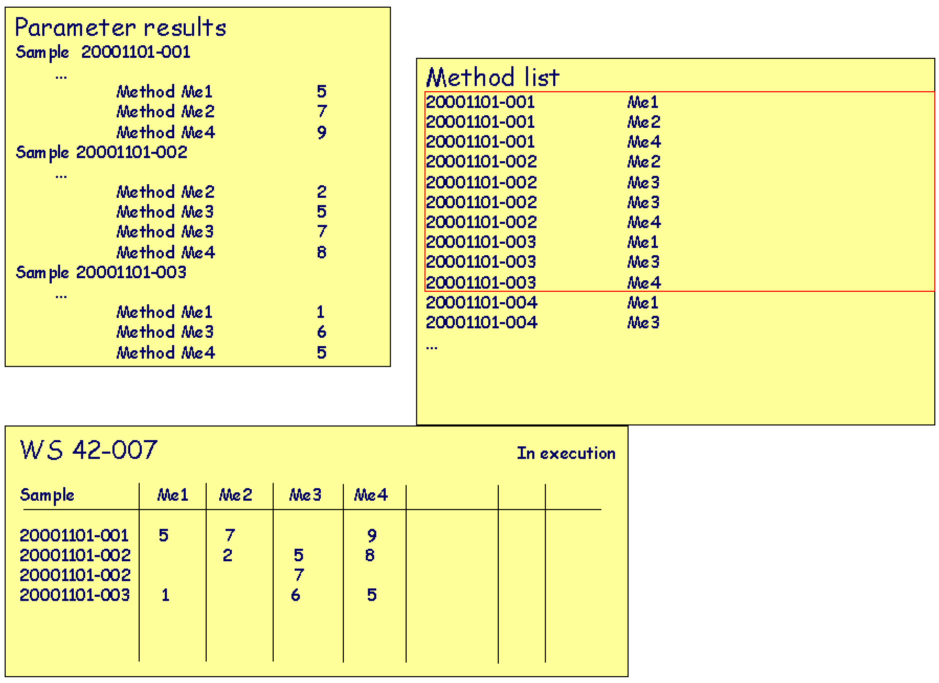
20001101-001	Mineral Water	11-Nov-2000
20001101-002	Mineral Water	11-Nov-2000
20001101-003	Waste Water	11-Nov-2000
20001101-004	Mineral Water	11-Nov-2000

WS 42-007 In execution

Sample	Me1	Me2	Me3	Me4
20001101-001	5	7		9
20001101-002		2	5	8
20001101-002			7	
20001101-003	1		6	5

- **wsAssignMeTask= <Method List Task>**: in this case, the group keys of the **Method List Task** (hidden on **Method list** window) and the matching **Method List Layout** will be used to present the user a method list for assignment of rows to the worksheet. As illustrated in the example below, the selection of 10 rows can result in an assignment of 4 rows in the worksheet.

The picture below shows how to assign methods to a worksheet from within a worksheet.



Parameter results

Sample 20001101-001		
...		
	Method Me1	5
	Method Me2	7
	Method Me4	9
Sample 20001101-002		
...		
	Method Me2	2
	Method Me3	5
	Method Me3	7
	Method Me4	8
Sample 20001101-003		
...		
	Method Me1	1
	Method Me3	6
	Method Me4	5

Method list

20001101-001	Me1
20001101-001	Me2
20001101-001	Me4
20001101-002	Me2
20001101-002	Me3
20001101-002	Me3
20001101-002	Me4
20001101-003	Me1
20001101-003	Me3
20001101-003	Me4
20001101-004	Me1
20001101-004	Me3
...	

WS 42-007 In execution

Sample	Me1	Me2	Me3	Me4		
20001101-001	5	7		9		
20001101-002		2	5	8		
20001101-002			7			
20001101-003	1		6	5		

In case both preferences are assigned to the **Worksheet Task**, the **wsAssignScTask** preference will be used.

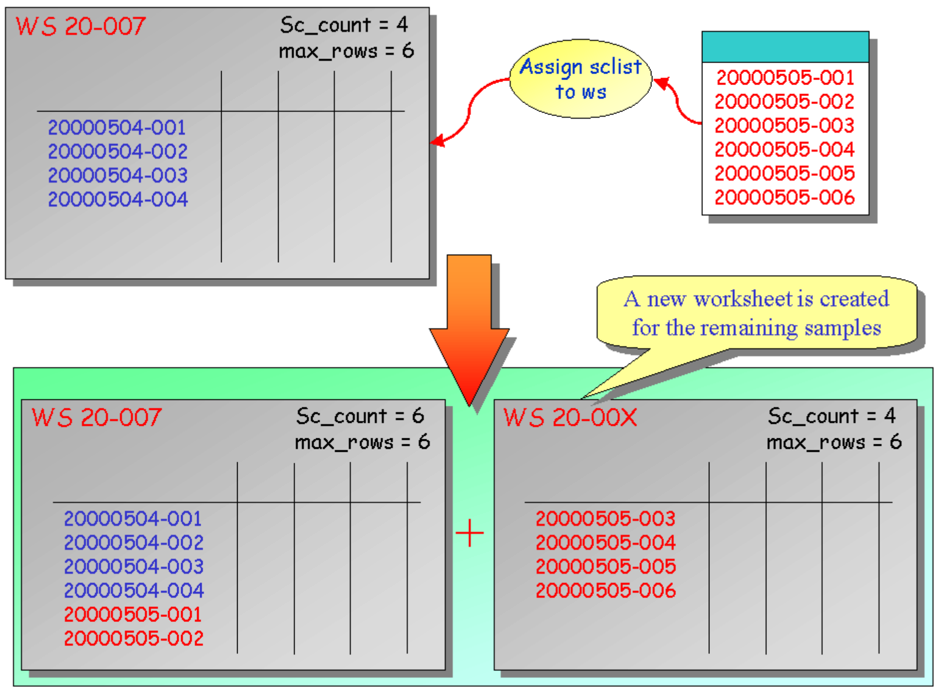
Samples or methods can be manually assigned to a worksheet from within a sample list or a worklist, respectively. The user has the option to either assign the samples/methods to an existing worksheet or create an entirely-new worksheet containing the selected samples/methods.

Automatic worksheet creation

Whenever a worksheet is completely filled, the system will immediately create a new worksheet of the same worksheet type to accommodate the remaining samples. The new worksheet will get a new unique ID.

Example: when a list of six samples is assigned to a specific active and open worksheet, the system will attempt to append as many samples as possible to the worksheet. However, when the worksheet only allows for the assignment of two more samples, a new worksheet of the same type will be created to accommodate the remaining four samples.

The figure below shows how to assign samples to a worksheet.



8.4.2 Worksheet Execution

The worksheet execution start date and the worksheet execution end date are automatically filled out by the system. These dates only depend on the execution start date and execution end date of the methods included in the worksheet.

Basically, the execution start date of the worksheet corresponds to the execution start date of the first executed method in the worksheet. The execution end date is determined by the execution end date of the last executed method in the worksheet.

8.4.3 Cancelling Worksheets

A worksheet can at any moment be cancelled. The cancel of a worksheet is trickled down to the method level: all methods included in the worksheet are cancelled as well.

Important

The samples contained in a worksheet are not cancelled when a worksheet is cancelled: these samples may still have other analyses assigned to them that are not contained within the worksheet.

8.4.4 Cancelling and Reanalyzing Objects from within a Worksheet

The operational worksheet is a set of objects that refer to other Unilab objects. As such, the worksheet is modified when one of the references is modified, not when one of the referenced objects is modified. The previous statement is very important to understand the behaviour of the worksheet object and the effect of reanalyzing and Cancelling objects from within the worksheet.

Reanalyzing methods in worksheet

Methods in a worksheet can be reanalyzed from within the worksheet as long as the worksheet status allows it. For methods, the re-analysis flag of the method is also included in the worksheet to make sure that the exact context is maintained. This is important in cases which a run has already been executed and, for some reason, later on, it is decided to reanalyze the obtained method result. This will most probably result in including the method in another run (=worksheet). When a method re-analysis is performed, the properties of the method in the original worksheet will not be modified. Hence, the worksheet will continue to display the original result of the run.

Note that assigning the same method with the same reanalysis flag to 2 different worksheets is not permitted.

Tip

If required, the propagation of a method re-analysis to the worksheet level can be handled in the method life cycle.

Cancelling objects in a worksheet

Cancelling a sample from within a worksheet will have the same effect as cancelling a sample from within any other Unilab application: all assigned objects (parameter groups, parameters, methods) are cancelled as well. However, when the worksheet itself is cancelled, only the methods that are part of the worksheet layout will be cancelled.

9 Equipment

9.1 Introduction

The majority of the analysis results in a modern laboratory are obtained through the use of analytical instruments. Many of these instruments are connected to the LIMS, sending the analysis results directly to the LIMS. Even if the instrument is not physically connected/linked with Unilab it is still possible to register/trace from which instrument the (manually-entered) result originated (for auditing purposes).

Full integration with mainstream LIMS functionality allows accurate and automatic use of calibration data in day-to-day operations of the laboratory and in the quality assurance of the results.

9.2 Equipment Properties

The generic object design of Unilab also applies to equipment.

Equipment life cycle

The state transitions in the equipment life cycle are primarily influenced by the warning level of the equipment. This is quite different from the type of events that are used to “navigate” through the life cycle of other objects.

Access rights on equipment

The system cannot prevent physical “access” to equipment, it can only disallow entering values for specific equipment (when the analyst has no access). This supports situations where the same method and type of equipment are used in different lab sections. Depending on the user profile, the right equipment will be selected for the user while executing the method.

However, authorization to execute a method is more often used to implicitly grant or deny access to equipment.

9.3 Equipment and Equipment Types

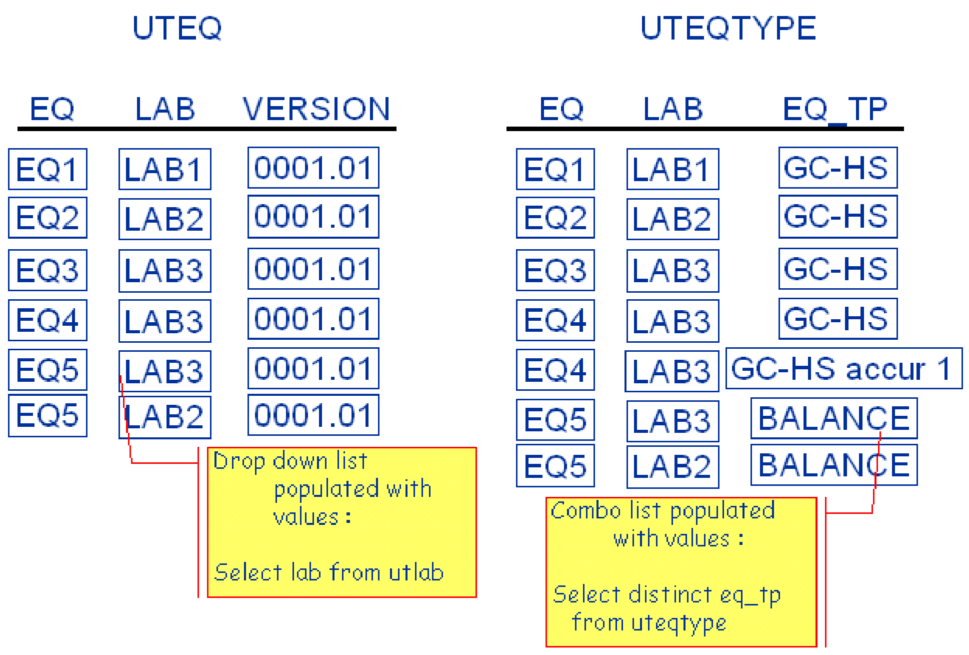
In order to standardize as much as possible, the LIMS must support generic method definitions that are useful in all plants. Ideally, the decisions about which equipment must be used to execute a particular method are delegated to the plant management level. The same holds for example for occasions where measurement ranges for the same method can differ from one plant to another, or when certain equipments are not available or replaced, etc..

Following these considerations, the concepts of a ‘Lab’ and an ‘Equipment Type’ are introduced in Unilab. On the corporate configuration level, methods are defined without specifying individual equipments but by using equipment types instead. As a matter of fact, only equipment types can be specified in the configuration of a method. When the configurator really wants to define ‘equipments’ rather than ‘equipment types’, the system must be configured in a sense that each equipment matches exactly with an equipment type. Consequently, for each distinct equipment, an equipment type must be created.

Furthermore, each equipment has an extra property ‘LAB’ that is part of the identification key, which links an equipment to a specific laboratory. By definition, a plant can have more than one laboratory. Also, an equipment can be linked to several equipment types.

On the operational level, when the method is executed, the system looks for the equipment matching with both the equipment type that was configured in the method and the lab context that is known in that context. The current lab context can be user-specific or computer-specific.

The figure below clearly illustrates the relation between equipments (table UTEQ) and equipment types (table UTEQTYPE). Note that both lab 2 and 3 can define their own EQ5 equipment as the identification key of an equipment is the combination of a ‘lab’ and an equipment type name.



UTEQ

EQ	LAB	VERSION
EQ1	LAB1	0001.01
EQ2	LAB2	0001.01
EQ3	LAB3	0001.01
EQ4	LAB3	0001.01
EQ5	LAB3	0001.01
EQ5	LAB2	0001.01

UTEQTYPE

EQ	LAB	EQ_TP
EQ1	LAB1	GC-HS
EQ2	LAB2	GC-HS
EQ3	LAB3	GC-HS
EQ4	LAB3	GC-HS
EQ4	LAB3	GC-HS accur 1
EQ5	LAB3	BALANCE
EQ5	LAB2	BALANCE

Drop down list populated with values :

Select lab from utlab

Combo list populated with values :

Select distinct eq_tp from uteqtype

As illustrated, lab3 can have 2 potential equipments, ‘GC1_HS for lab3’and ‘GC2_HS for lab3’, in the figure named as EQ3 and EQ4. By this, corporate headquarters does not decide on the method level which exact equipment to use, but forces the use of an equipment of type ‘GC_HS’.

9.4 Regulatory Compliance

The analytical instruments in the laboratory are subject to regulatory compliance, which requires keeping certain information on the instrument and the establishment of tedious calibration and maintenance procedures.

For each analytical instrument, the system can maintain all supplier and maintenance related information. Equipment information covers (according to ISO17025):

- ID and name of equipment
- Name of supplier
- Type, description and serial number of instrument
- Date and conditions of reception and first use
- Current location and operating conditions
- Detailed information about the equipment
- Normal maintenance (date and description + user ID)
- Repairs (including identification of replaced parts)
- Calibration scheme and used standards

This information is stored in the equipment definition properties. Documents containing the written procedures (SOP) are available on line, thus assuring regulatory compliance. This is implemented through attribute services.

9.5 Intervention Rules

To implement QA rules properly, equipment intervention procedures must be setup. Interventions include every action that needs to be taken on an instrument to assure that it operates on a daily basis within the established limits. Examples of interventions are calibration activities, maintenance, repair and cleaning. The properties of an intervention include:

- The intervention description
- The standard operating procedure for intervention execution
- The intervention frequency
- The current status of the intervention
- An audit trail of each individual intervention

Usually, several different types of interventions are required for one instrument. Therefore, the system allows scheduling multiple interventions for a single instrument.

9.5.1 Intervention Properties

Identification

In addition to the name and the description, the intervention properties also include an indication of the cost and the required time to execute the intervention. This information can be included in equipment survey reports.

SOP

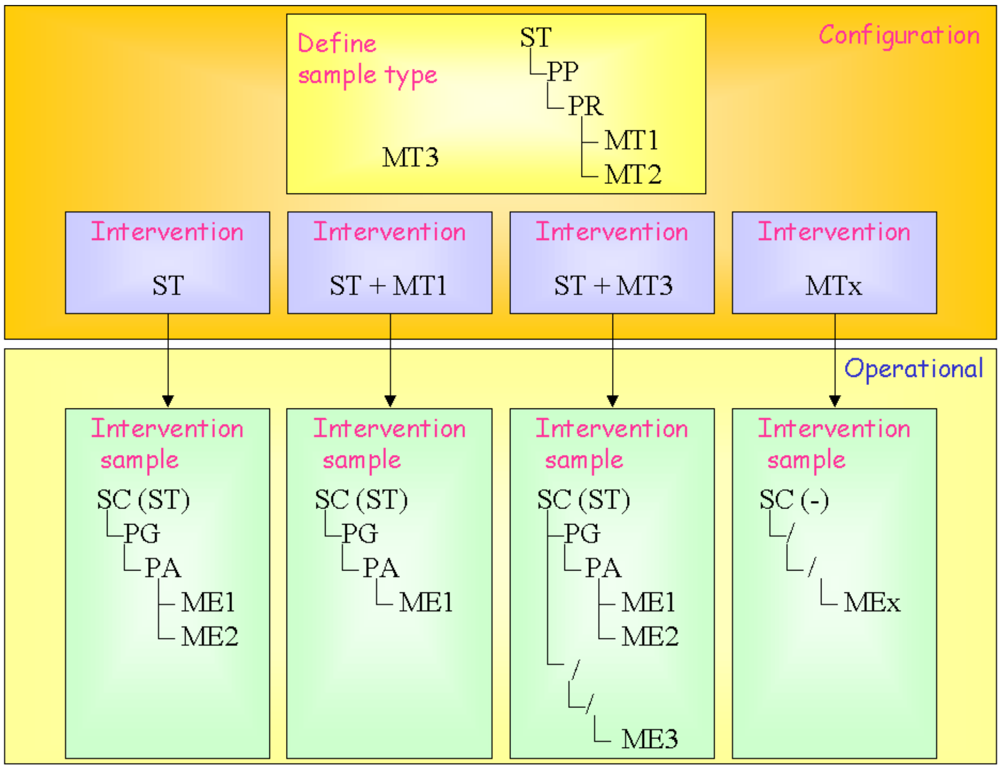
A standard operating procedure describing how the interventions should be executed can be linked through an attribute service. When performing a calibration or maintenance, an intervention sample is created to execute the equipment calibration method(s).

Intervention sample or method

For each intervention, an intervention sample type and/or an intervention method can be defined.

Intervention sample versus method

The figure below shows the different possibilities on how Unilab handles the intervention samples and/or methods.


Define sample type
ST
PP
PR
MT1
MT2
MT3
Configuration
Intervention ST
Intervention ST + MT1
Intervention ST + MT3
Intervention MTx
Operational
Intervention sample SC (ST) PG PA ME1 ME2
Intervention sample SC (ST) PG PA ME1
Intervention sample SC (ST) PG PA ME1 ME2 / / ME3
Intervention sample SC (-) / / MEx

The sample type defined for the execution of a specific intervention can hold specifications for proper equipment evaluation. These specifications are set as parameter specifications in the sample type. The evaluation of the corresponding parameter result against these specifications triggers the necessary state transition in the equipment life cycle.

In addition to the intervention sample type, an intervention method can be defined. Any configured method can serve as an intervention method. However, only methods specifically marked as calibration methods can be executed on inactive equipment.

If the method was not yet assigned to the intervention sample type, an extra parameter group will be added to the list of parameter groups of the corresponding intervention sample. This parameter group "\" does not contain any parameters. Only at the method level, the method description is filled out.

The option also exists to only define an intervention method. In this case, intervention samples will also be created, but they will not belong to any sample type.

Creation of intervention sample

Whenever an intervention sample is created, the system will automatically assign the following two group keys to the sample:

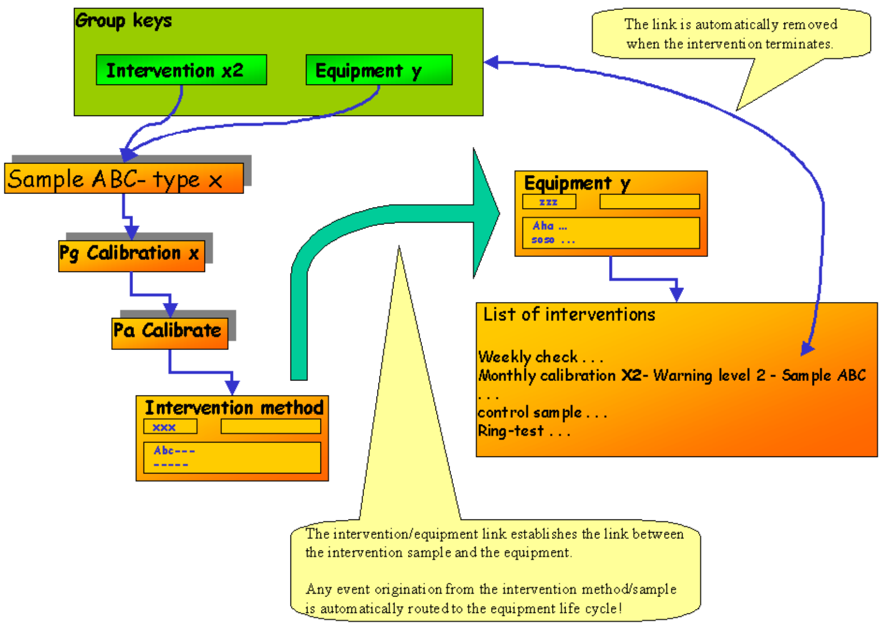
- **Intervention4eq**
- **Intervention4lab.**

Note Group key **Interventionrule** is also assigned automatically to the intervention sample.

Both sample group keys can be used to find out which interventions still need to be done. The (sample) group keys can also be used to fetch a list of intervention methods.

These two intervention group keys are automatically removed (from the sample) when the intervention is terminated (= the status of the intervention is – either manually or automatically – modified to warning level 0 or 4).

The figure below shows the concept of intervention group keys.


Group keys
Intervention x2
Equipment y
The link is automatically removed when the intervention terminates.
Sample ABC- type x
Pg Calibration x
Pa Calibrate
Intervention method
xxx
Abc---

Equipment y
zzz
Aha ...
soso ...
List of interventions
Weekly check . . .
Monthly calibration X2- Warning level 2 - Sample ABC
. . .
control sample . . .
Ring-test . . .
The intervention/equipment link establishes the link between the intervention sample and the equipment.
Any event origination from the intervention method/sample is automatically routed to the equipment life cycle!

9.5.2 Intervention Frequency

The majority of interventions are executed on a regular basis. The intervention frequency can be:

- Time based
- Usage count based

- Custom function based.

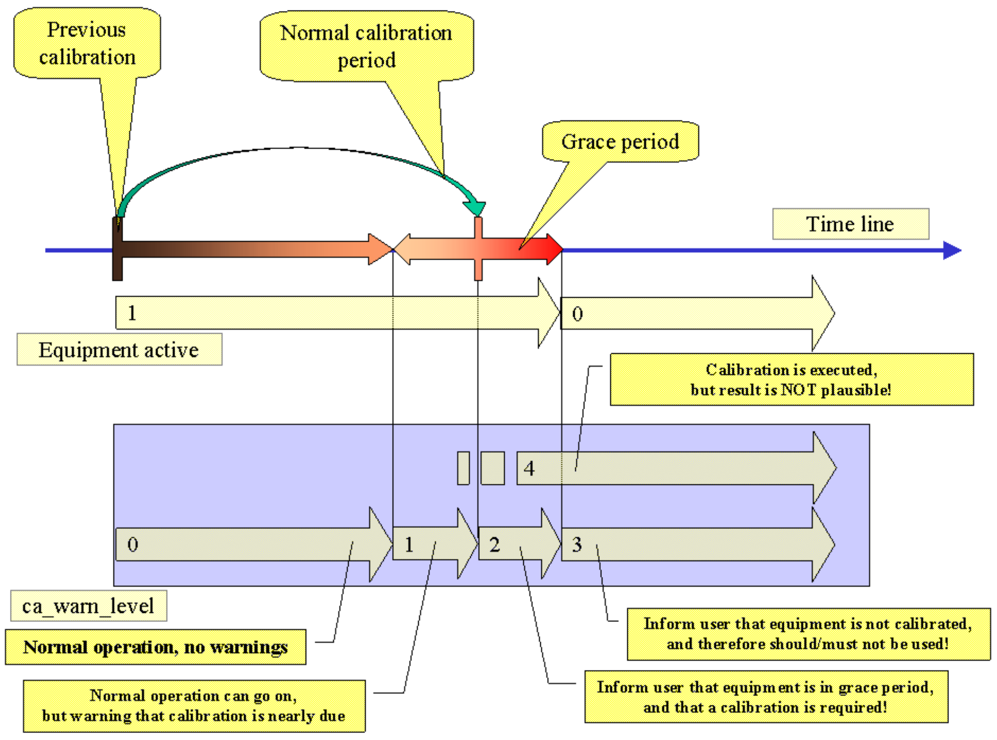
Sometimes, a grace period is allowed for an intervention This is a period after a required intervention, in which the equipment can still be used, but extra user attention is required (a message will be forwarded to the user, explaining the current status of the equipment). The grace period can also be used to define a period before the next intervention in which the user is warned that an intervention is almost due. However, normal operation of the equipment is still possible. A grace period can be specified as a time period or as a number of analyses that can be performed on the equipment.

Sometimes an intervention must be performed immediately. For example, this is the case when something extraordinary, such as physical damage or electrical overload, occurs to the equipment. At any time, the analysts can manually create/start an intervention. The manual logon of interventions has no influence on the intervention frequency.

9.5.3 Intervention Warning Level

Each intervention rule has its own warning level. The warning level of an intervention gives an indication about the status of that intervention. The warning level of the instrument corresponds to the highest warning level of its interventions.

The figure below shows the different warning levels of an intervention.


Previous calibration
Normal calibration period
Grace period
Time line
1
0
Equipment active
Calibration is executed, but result is NOT plausible!
4
0
1
2
3
ca_warn_level
Normal operation, no warnings
Normal operation can go on, but warning that calibration is nearly due
Inform user that equipment is not calibrated, and therefore should/must not be used!
Inform user that equipment is in grace period, and that a calibration is required!

Warning level **0** indicates that the last execution of the intervention was successful. Normal operation is possible, no warnings are forwarded to the user. In the Warn upfront grace period, the warning level is **1**. Normal operation of the equipment can go on, but the user will be warned that an intervention is due soon. Warning level **2** indicates that the equipment is in the grace period. In case the grace period expires without an intervention being executed, the warning level for that intervention is raised to **3**. A user trying to use the equipment will either get a message that the equipment should be calibrated (in case he/ she is not authorized to perform the intervention) or the intervention box appears (when he/she is authorized to use the equipment). Either way, the equipment can no longer be used.

The successful completion of an intervention will reset the warning level (for a specific rule) to zero. When the intervention is not successful, the warning level (of the corresponding intervention rule) will be raised to **4** and, hence, the warning level of the equipment will also be raised (to **4**).

The systematic and automatic evaluation of intervention rules (periodically for time based rules and “on equipment use” for count based and “custom function” based rules) guarantees that the warning level will always be up-to-date.

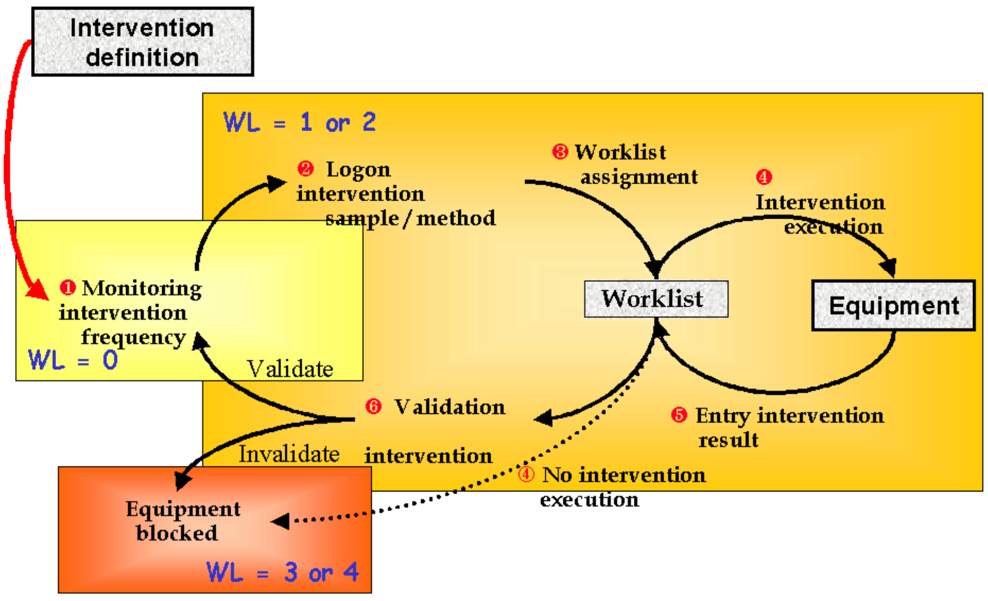
9.5.4 Intervention Logon and Execution

The procedure of intervention logon and execution is displayed in the picture below. After an intervention frequency expires, the system will automatically log on a sample of the specified intervention sample type (and assign the intervention method to that sample – if specified).

The “intervention equipment” and the “intervention rule” (sample) group keys are automatically assigned to the intervention sample; it is not even necessary to define any group-key assignment rule(s). In the **utscgkintervention4eq** and **utscgkinterventionrule** group keys are automatically created as part of the standard installation.

These group keys can be combined in an intervention task. In this manner, the analysts, responsible for performing the intervention, can easily retrieve a list with all intervention samples.

The figure below shows an intervention execution.


Intervention definition
WL = 1 or 2
❷ Logon intervention sample / method
❸ Worklist assignment
❹ Intervention execution
❶ Monitoring intervention frequency
WL = 0
Worklist
Equipment
Validate
❻ Validation intervention
❺ Entry intervention result
Invalidate
④ No intervention execution
Equipment blocked
WL = 3 or 4

9.5.5 Results of an Intervention

The results of an intervention can be obtained in two different forms:

- The sample status (which may be translated into the equipment status itself)
- A set of equipment constants (e.g. the coefficients of the calibration curve).

Equipment constants are all equipment-specific constants required for the normal operation of the instrument. These can be the coefficients of the calibration curve(s), correction factors, etc.

For auditing purposes, it is required to keep track of historical values of equipment constants. Therefore, a history logging is kept. The column **intervention** in the history table indicates which intervention was used to determine the corresponding equipment constant. It is assumed that under normal circumstances, these constants are measured/updated by executing a "calibration" method (scheduled according to a specific intervention rule). However, these equipment constants can also be edited manually. In the case of manual changes, the "intervention" rule will be set to **"MANUALLY ENTERED"**.

The historical values can also be presented in trend charts, which are defined in the Configuration application.

9.5.6 Intervention Validation

An executed intervention can both be validated manually or automatically. Automatic validation is handled through the life cycle used for the intervention sample. The sample type defined for the execution of a specific intervention can hold specifications for proper equipment evaluation. These specifications are set as parameter specifications in the sample type. The evaluation of the corresponding parameter result against these specifications triggers the necessary state transition in the equipment life cycle.

In case an intervention is validated, the warning level is set to 0 again. However, interventions can also be invalidated. This is for instance done when the intervention has not been executed properly. The warning level of the intervention is then raised to 4. Hence, the overall warning level of the equipment will be raised to 4 as well).

9.5.7 Traceability

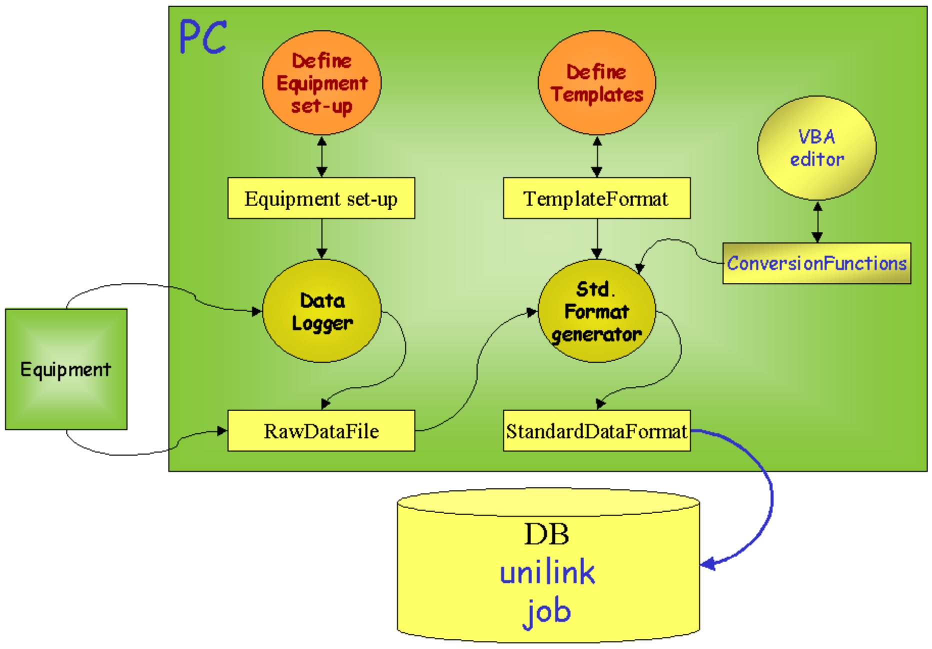
It is also possible that the user schedules reference samples to verify that the equipment is still operating correctly (regardless the calibration frequencies). The result of such a measurement could be an immediate (re-)calibration! Trends of such results (over time) are very important to evaluate the operation of the equipment. Therefore, the equipment constants are logged for easy trending and SPC. Equipment constants will be stored both as floats and as strings (similar to the storage of parameter results). Equipment-constant trends are defined in the Configuration application, and are assigned to a certain equipment in the **Define Equipment** application.

9.6 Interfacing Unilab with Laboratory Instruments

Unilab is used to process and centralize analysis results coming from different laboratory instruments . This implies communication between the LIMS and these instruments. Unilab receives results from the instruments and enters these results into the database. Unilab is also able to trigger analyses on an instrument and receive the results.

9.6.1 General Overview

Unilab provides functionality to interface with external systems. External systems can both be analytical instruments or other applications used in the laboratory or enterprise. The figure below shows how an external system can be interfaced to Unilab.



The process contains two steps:

1. Capture of data coming from the external system and their preparation for processing
2. Processing of data and their insertion into the Unilab database by means of a UNILINK job.

9.6.2 Capture and Standardization

Depending on the external system, there are two ways to capture the raw data. Many applications or analytical instruments generate a data file (which is saved in a specific directory). This data file, which is often an ASCII file, is picked up by Unilab. The raw data is then parsed and saved in a standardized format by the standard format generator.

Unilab not only processes data files (coming from any equipment), it also supports capturing data directly from a (number of) serial port(s) local on the client. The data logger generates a raw data file, which is again parsed and saved in a standardized format.

In order to interpret a raw data file coming from an instrument, a description of the relevant data parts must be given. This descriptive file must also contain information about how to save the data.

The data file template allows establishing a map between the contents of a raw data input file and the standardized output file. Any number of data templates can be defined.

Quite a number of fields extracted from a raw data file need to be converted and or formatted before they can be stored in the database. For example, this includes:

- Reformatting of dates
- Concatenation of fields
- Splitting and re-arranging of sub-fields

The conversion can be handled by Visual Basic conversion functions. Some of the conversions have to be implemented by means of lookup tables on the client. These conversions are equipment-specific. Therefore, specific conversion tables and conversion functions are kept on each client.

The following items must be defined for all equipment:

- The input file sources (COM port or directory files)
- The output file destinations

9.6.3 Processing and Insertion into the Unilab Database

Once the data that was captured is saved in a standardized format, it is processed and inserted into the appropriate Unilab database tables. This is done by means of a UNILINKjob, which can be customized, but its default implementation handles the described **StandardDataFormat**.

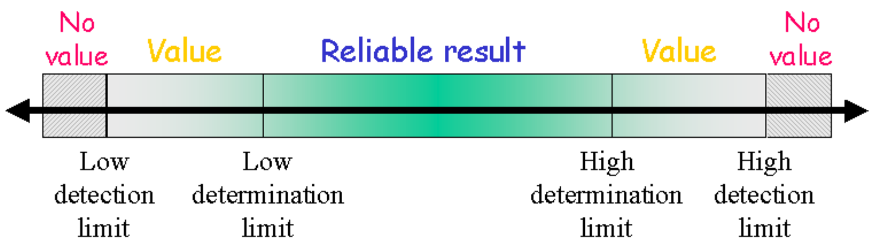
9.7 Measurement Ranges

A single laboratory instrument can be used to detect a number of components in a sample. However, each instrument has its physical and practical limitations for the analyses of the various components. These limitations are defined through measurement ranges. The equipment has different measurement ranges for each component that can be measured with the instrument.

9.7.1 Definition of Measurement Ranges

Two kinds of measurement ranges are used in Unilab: detection limits (low detection limit and high detection limit) and determination limits (low determination limit and high determination limit). The exact definition of each of these measurement ranges slightly varies according to different laboratories. However, the following definition is commonly accepted and applies to Unilab. The detection limits define the physical limits that can be handled by the instrument. Below the low detection limit of a specific component, the instrument is not sensitive enough to detect the component. Above the high detection limit, the instrument becomes saturated. Determination limits define the zone in which the obtained results are reliable. A result outside the determination limits, but within the detection limits, indicates that the specific component is present in the sample, but the actual value is not “accurate”.

The figure below shows the detection and determination limits of an instrument for a component.



Important

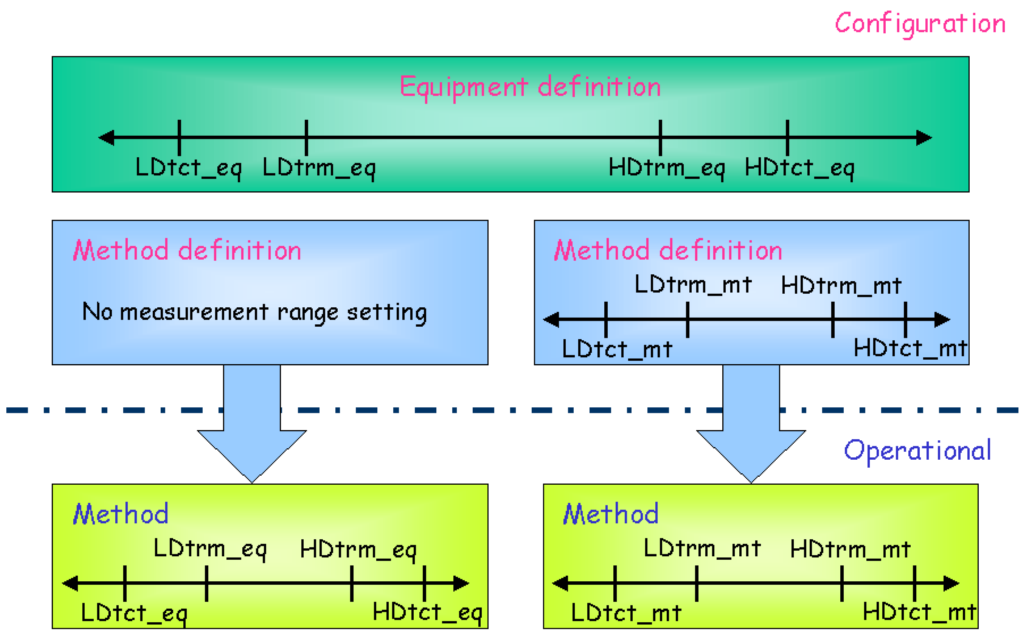
These definitions are subject to interpretation. Unilab allows for different interpretations, as long as the detection range encompasses the determination range (the low detection limit is lower than the low determination limit, and the high detection limit is higher than the high determination limit).

Measurement ranges of an instrument depend on the component that is measured and on the method that is applied to determine the component.

Often there are interfering compounds present in real samples. These compounds influence the measurement ranges of the equipment. One might say that the measurement ranges defined at the equipment level only apply for the concentration of a component in ideal samples without any interfering matrices (e.g. pure water).

Consequently, Unilab allows for the definition of measurement ranges at the method level. When present, measurement ranges at the method level always overrule measurement ranges at the equipment level.

The figure below shows how measurement ranges at the equipment level are overruled by measurement ranges at the method level.



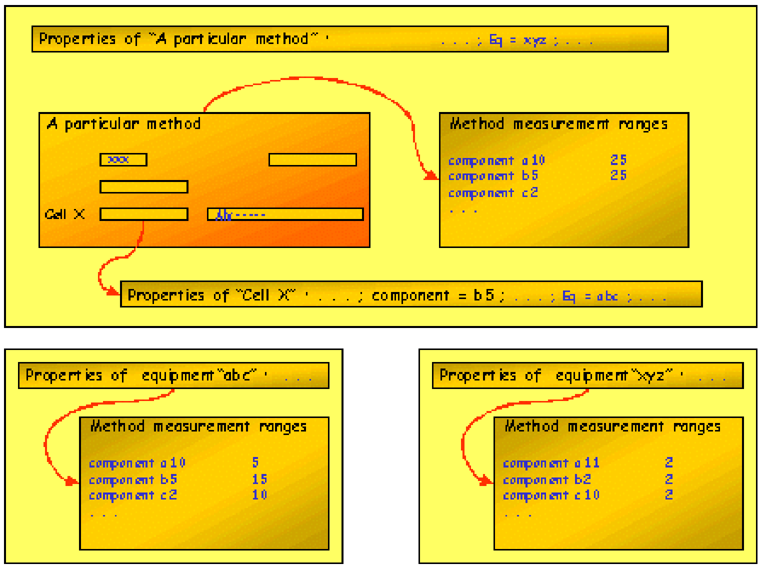
9.7.2 Checking the Measurement Ranges

Measurement ranges apply to values in method cells. After entering method-cell results, the client application will always (first) apply the normal conversion and formatting rules. Then the measurement ranges are checked.

The method-cell values are linked to the correct measurement ranges through the definition of a component within the method-cell properties.

If the measurement ranges for the specified component are defined at the method level, these will be used to format the result accordingly. This is completely independent of any equipment specification at the method and/or cell level.

The figure below shows the measurement ranges on equipment and on method level.


Properties of "A particular method" ' . . . ; Eq = xyz ; . . .
A particular method
xxx
Cell X
Abc - - - - -
Method measurement ranges
component a10 25
component b5 25
component c2
. . .
Properties of "Cell X" ' . . . ; component = b5 ; . . . ; Eq = abc ; . . .
Properties of equipment "abc" ' . . .
Method measurement ranges
component a10 5
component b5 15
component c2 10
. . .
Properties of equipment "xyz" ' . . .
Method measurement ranges
component a11 2
component b2 2
component c10 2
. . .

Otherwise the system will attempt to retrieve the measurement ranges from the equipment corresponding with this cell. If there is no equipment for the current cell, the equipment at the method level (if any) will be used as an alternative.

Tip

The actual formatting rules when measurement ranges have been specified are discussed in the chapter on Formatting and Conversion of Results.

10 Requests and Request Types

10.1 Introduction

A request consists of a number of samples that have to be processed in a grouped way. In addition, processing requires a work flow (life cycle) and information of its own. A fast and accessible link should be maintained between samples. Examples of requests are:

- A batch of samples arriving together in the laboratory
- A complex sample from which several sub-samples are taken
- A complaint filed by a customer
- A research project.

Types of analysis requests

Roughly, analysis requests can be subdivided in two categories. The first category includes requests for which the test plan has already been established beforehand. It is perfectly known which and how many samples are to be taken within the context of the request and what analysis are to be executed on these samples. Most requests logged in a QA or in a production environment belong to this category. An example of such a request is a production batch: the number and type of samples, the sampling date (time) and the test plan are established on forehand.

In a service laboratory, most analysis requests that are logged try to seek answers to specific questions. It is never known on forehand what will be the nature of the request/question, what samples are to be taken or what analysis are to be done. Even with samples of the same sample type, the nature of the requests can be very different. These requests constitute the second category of requests.

Typical examples of such requests are complaints: the samples to be taken or the analysis to be assigned depend on the nature of request. Also research projects belong to this second category of requests.

Requests in Unilab

A request represents an additional grouping of samples on top of the group-key mechanism. Each sample in a request can have a different sample type. Each sample has its own info cards and follows a specific life cycle. The request itself also has its own life cycle. The request's life cycle status largely depends on the statuses of the individual samples. Administrative information for the entire request is kept on its own info cards. This eliminates the need to duplicate a lot of information across samples that belong to a particular request.

Further grouping of requests can be done based on request characteristics such as request category or priority. This grouping is implemented by means of group keys. Group keys also provide a fast and efficient way to retrieve a list with requests having common characteristics.

To minimize data entry during the process of logging in requests, the properties of requests with common characteristics are defined in a request type.

10.2 Request Properties

The majority of request properties are inherited from the request type to which it belongs. However, Unilab also offers the possibility to manage certain properties at the request level.

10.2.1 Identification of Individual Requests

Each analysis request is uniquely identified by means of a request code. The code generation algorithm for the request code is defined by means of a unique code mask. The unique code mask used for requests of a certain request type is included in the request-type definition. The request code is the unique identifier for the request in the database. This also implies that, once the request has been created in the database, the request code cannot be changed anymore.

The properties of a request always include a reference to its request type and its description. It is also possible to define a second description (per request type). The second description can only be used for reporting purposes. For example, this second/alternative description is displayed on analysis certificates (which can be sent to external parties – in other countries).

10.2.2 Request Creation and Processing Details

Many of the request property values are supplied at request creation. Some of these provide details on the creation process itself, such as the creation date (the date the request was logged into the database) and the ID of the user who created the request. A field also holds the sampling date, e.g. the date when all samples of a particular request were taken. By default, the sampling date is set equal to the creation date. However, the sampling date can always be modified.

Some request properties are inherited from the request type. This is the case for the priority and the standard operating procedure (SOP) that describes how the request should be handled. Although these properties are inherited from the configuration level, they can be modified at the operational level.

Processing details of the sample are first of all provided by its current status in its life cycle. The life cycle used is set at the request-type level. Authorized users can always assign another life cycle model to the request (although changing the life cycle is rather exceptional). The start of the analyses (entering parameter results for the samples of the request) and the eventual completion of the analyses are also filled out automatically.

A due date can be entered as well: this is the date on which all analysis for the request should be finalized. This due date can be used in the request life cycle to trigger a state transition in case the due date is near.

10.2.3 Optional Date and Time Info

Besides the sampling date and the request creation date, other dates might provide critical information on the requests. An example is the date on which the certificate was generated. For each request, up to five additional dates can be stored. The labels of the optional date fields are set system wide through the system settings **LABEL_rq_DATE1** till **LABEL_rq_DATE5**.

The additional date info can be entered manually in the request property sheet. Their value can also be set as a default value for an info field in one of the request's info cards. This default links the info field to an optional date field enabling the value to be filled in by supplying the info field.

The value of the optional date info fields can also be set by an action triggered by a state transition in the life cycle.

Example: The label for **date 1** is set to **Certificate date**. In this field, the current date is supplied by an action on the state transition from **Validated** to **Reported** in the request life cycle. The state transition itself could be triggered by the generation of the certificate of analyses for the involved request.

10.2.4 Barcode Labels

Requests and samples belonging to a request are often labeled on arrival in the laboratory for identification purposes. For example, the label can be a barcode label, used to speed up data retrieval for the request or to identify the link between the individual samples belonging to the same request.

10.2.5 Cancelling or Deleting Requests

When a request is cancelled, all objects at the lower hierarchical levels (info cards, samples, parameter groups, etc.) are also cancelled. The worklist assignment rules are re-evaluated for assigned methods.

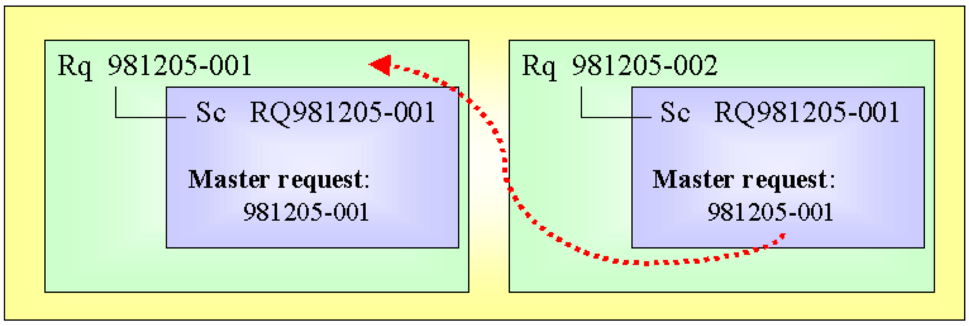
Deleting a request can only be done for requests that are not active. For deleted requests, all references in the database are deleted as well. Whether or not the samples belonging to a request are deleted on request deletion is controlled at the system level through the system setting **RQDELETECASCADEONSC**.

10.2.6 Request Samples

Samples belonging to a request

An analysis request can have one or more samples assigned to it. Viceversa, one sample can belong to several requests. However, only one request to which the sample belongs is set as the master request of that sample. The ID of this request is kept in the sample properties.

The figure below shows the master request of a sample.



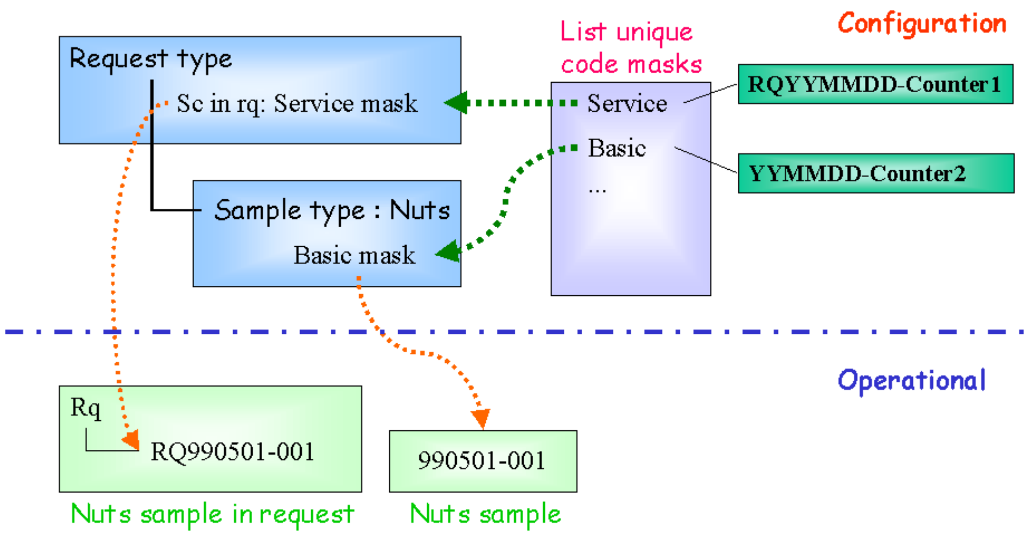
Identification of request samples

Samples belonging to a request can be identified in several ways. First of all, they are identified through the unique ID that has been assigned to the batch as a whole. This ID is implemented by means of a request code.

The samples belonging to the request have their own unique ID. This ID often makes reference to the request itself. This is implemented by means of a sample code mask.

In many cases, a clear distinction must be made between samples of the same sample type that belong to a request and those that do not belong to a request. Therefore, the used sample code mask is set on the request-type level. This setting always overrules the corresponding setting on the sample-type level.

The figure below shows the unique code generation for a sample.



The samples belonging to the same request are often numbered sequentially. This can be implemented through special fields inside the sample code masks. The field **rq seq1** is used to count the number of samples in a request, starting from 1. The field **rq seq0** is used to count the samples in a request, starting from 0.

Example

The following code mask is used for the samples of a request:

```
RQ-{rq}-{rq seq0}
     With rq = request code.
The samples for request 20000303-001 will have following
sample codes:
RQ-20000303-001-000
RQ-20000303-001-001
RQ-20000303-001-002
RQ-20000303-001-003
…
```

10.2.7 Dynamic Assignment of Samples to a Request

Samples can be added dynamically to existing requests. The responsible user should have the appropriate access rights to do so.

Assigning existing samples to a request

The assignment of existing samples to a request is controlled through the **Allow any Sample Code** flag on request-type level. When this flag is set, any existing sample can be assigned to the current request.

When this flag is not set, no existing samples can be assigned to the current request.

Assigning new samples to a request

Newly-created samples from within a request list are always assigned to the current request (this is the request that has the focus on the moment that the sample creation functionality is called).

The list of sample types displayed to create new samples is controlled by the setting of the **Allow any sample type** flag on request-type level. When this flag is set, the user will get the list of all active sample types for the creation of samples. When the flag is not set, the user only gets the list of sample types that have been explicitly assigned to the corresponding request type.

10.2.8 Request Analysis Details

The request analysis window displays all samples for a request. For each sample, the following data can be displayed and filled out:

- Parameters belonging to a specific request-type parameter profile
- Sample-code info fields

10.3 Request-type Properties

A request type is the template definition for specific types of requests used to fasten and simplify the creation of new requests. In addition, some settings are provided to control the properties of the request type itself.

The request-type property sheet includes a number of settings that define the properties and the behaviour of requests at the operational level. Each of these properties is set manually at the configuration level.

These settings include:

- The default life cycle model for requests of this type
- The code mask used to generate the request codes
- The standard operating procedure describing how to handle requests of this type
- The default priority for these requests
- The label format used for the requests.

For reasons of security and the audit trail, it is sometimes required that the user ID be confirmed on request creation. This feature can also be set at the request-type level.

Request group keys

Group keys can be assigned to a request type. However, these group keys are not used inside the **Configuration** application itself (the list of request types can be searched using standard search functionality).

The group keys defined on the request-type level are used during the new request creation process to make the identification of requests in the **New request** window easier.

The request-type group keys can be inherited by the requests belonging to this request type as well. In this way, request-type group keys can be used to restrict the request list to a meaningful set in the request list.

Tip

For more details on the inherit mechanism for group keys, please refer to the chapter on Inherit Group Keys in **SIMATIC IT Unilab - Concepts Guide Part 3**.

Linking sample types to a request type

A number of sample types can be assigned to the request type. For each sample type, it can be specified how many samples of that sample type have to be created at request creation and what the assignment frequency should be. Note that the configuration settings can be overruled at the operational level.

Info profiles of a request type

Information for a request can be kept in info cards. Therefore, the corresponding info profiles should be assigned to the request type. For each of the info profiles, the assignment frequency can be set.

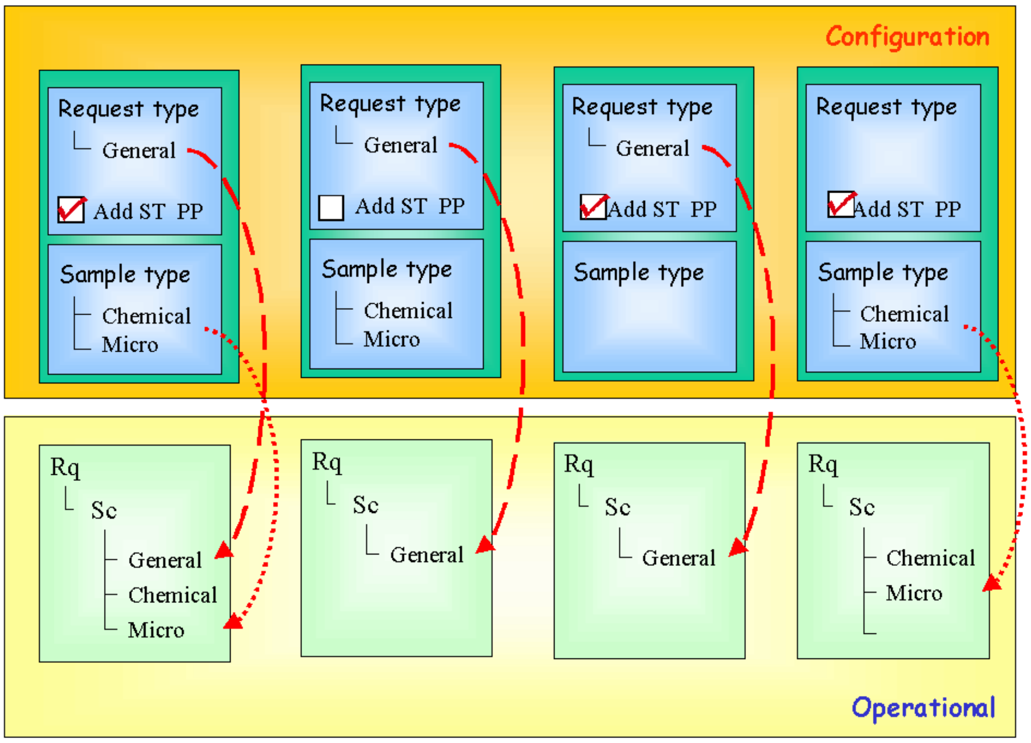
Example: When a customer files a complaint, information should be stored about the name of the customer, the nature of the complaint, etc. This information can be entered in an info card assigned to the request. On the link “request type – info card”, attributes can be defined.

Parameter profiles

A request type can have a list of parameter profiles assigned to it. This list can be used as a basis for assigning parameter groups to the request samples at the operational level.

Whether or not sample-type parameter profiles should be assigned to samples at request creation can be controlled through the **Add sample type parameter profiles** flag. When this flag is set, the parameter profiles assigned to the sample types of the current request type are attached to the samples (on top of the parameter profiles assigned in the parameter profile template of the request type itself).

The figure below shows the different possibilities in using request-type parameter profiles and sample-type parameter profiles.



10.4 Request Creation

Generation of the request

For each request that is logged on, the system automatically generates a unique code, based on the unique code mask of the corresponding request type.

Some of the request properties are supplied at request creation. This includes the creation date, the ID of the request creator, and the sampling date is filled out automatically. The default value of the sampling date is controlled through the **RqCreateSamplingDate** preference. In contrast to the creation date, the default sampling date can always be modified once the request creation process has terminated.

Creation of request details

The procedure applied to create the request details (samples and test plan) depends on the request category.

For requests, such as a production batch, where the test plan can be established on forehand, these settings can be defined on the configuration level. The request type then includes:

- The type and number of samples to be created within the request context
- The parameter groups to be assigned to the request samples and their assignment frequency.

Upon request creation, the corresponding request-type settings are applied.

For requests, such as complaints or research projects, the required analyses are to be assigned at request creation.

The creation of request details is controlled through the **rqCreate** preference. This preference can also be set as attribute on the request-type level.

Group-key initialization

At request creation, the appropriate group keys will be initialized (according to the group-key assignment rules). The group keys and their values, specified at the request-type level, can also be inherited by the requests.

Creation of request info cards

By default, info cards are created during sample creation based on the specified assignment frequency. Whether or not the infocard are created at sample creation can be controlled through the preference setting **rqCreateIc**.

Creation of request samples

At request logon, the samples belonging to the request are created, based on the defined request-detail creation strategy. However, this can slow down the request login, especially when large numbers of requests or requests with a large volume of samples are logged in at the same time. Moreover, the samples that need to be taken for a specific request can depend on the information filled out in the request info card(s).

Sample creation can be controlled through the **rqCreateSc** preference.

11 Studies and Protocols

As of SIMATIC IT Unilab 6.1, dedicated support for **stability** testing is provided. Stability testing is a routine procedure performed on drug products, active ingredients and excipients in order to understand the long-term effects of the environment on the drugs. It is used at various stages of product development. In the early stages, accelerated stability testing (high temperatures and relative humidity) is used to evaluate the stability of the drug and to determine what types of degradation products may be found after long-term storage. Testing under normal conditions (those recommended for long-term shelf storage) can be used to determine a product's shelf life and expiration date. In these types of studies, the product is analyzed at predetermined intervals for various parameters.

To define the functionality of the **Stability** module, the functionality is divided into 2 major parts:

- A configuration part, which consists of:
  - Condition sets
  - Locations
  - Protocols
- An operational part: a **Stability Study**

  The operational blueprint of the protocol, also containing locations next to the time point and condition sets.

  A detailed description on the objects used for stability testing can be found in the manual **SIMATIC IT Unilab – The Stability Module**.

12 Documents

12.1 General information

Generally speaking, researchers often use documents written in MS Word, Excel or other formats (for example, .ppt, .xls) to capture and convey the actual information of the context within which they are working. While providing flexibility, such documents in themselves do not offer sufficient access control nor audit information.

From the Unilab standpoint, a document is conceived as a data object that combines the characteristics of both configuration and operational objects. In light of this, the concept of "document" in this context does not contemplate having a configuration object from which its operational counterpart is generated in run-time. This holds for both documents created from scratch, as well as pre-existing documents that provide additional information to the user.

A document need not necessarily be text, but may also be a set of slides, captured images, flowcharts, etc.

When a document is created, the user has the possibility of deciding whether it will have a life cycle or not. This has impact on how changes to a document's content and/or properties can be made and become effective. For more information, see Documents with a life cycle and Documents without a life cycle.

Likewise, versioning also depends on whether a document has been created with a life cycle or not. For more information, see Document versioning.

Access control and audit information are provided by storing the documents as BLOBs (Binary Large Objects) in the DB. Any type of document can be stored in the DB.

Each client can use its own "viewer", specific to the type of format in which the document is written/provided (for example, text documents in .pdf format can be viewed via Acrobat Reader).

The Document Handling environment consists in a single workspace for managing documents.

12.1.1 Document properties

Like all other data objects, although being not classifiable as either a configuration object or an operational object, a document has properties. The majority of these properties are similar to those present in other data objects: they concern the document's identifier and description, as well as details concerning the document's author and time of creation.

Likewise, access rights permit identifying which users can access the data domain(s) where the document is saved in the database, whereas custom attributes, group keys and audit trail details may also be present.

But a major difference concerns the presence or absence of a life cycle for a document. This factor affects both editing and versioning. As a result, a document’s properties may or may not include details regarding its life cycle and current status (see Default Life Cycle for Documents - Available States), as well as specific info regarding its versioning (i.e. indication as to whether the document is currently checked-in or checked out). For more information, see Document versioning and Check-Out/Check-In procedure.

12.1.2 Document versioning

Versioning depends on whether a document has a life cycle or not.

If a document….	**Then its versioning is….**
Has a lifecycle	Decided by the user (i.e. it is the user who will decide whether or not to generate a new version of the document on which he/she is working).. There exists the possibility of creating both major and minor versions.
Does not have a lifecycle	Automatic, on the basis of the Check-Out/Check-In procedure (i.e. a new version is automatically created each time the modified document is checked in).

12.2 Documents with a life cycle

Editing documents that have been created with a life cycle is a straightforward procedure, as it does not require the Check-Out/Check-In operation in order for the changes made to become effective.

12.2.1 Default Life Cycle for Documents - Available States

The table below illustrates the various states that may be assumed by a document that has been assigned the default life cycle.

The original files of such documents may be overwritten with changes by the user, provided that:

- the document is in an editable status;
- the user possesses the appropriate FARs and Access Rights to make modifications.

Note Documents with a life cycle other than the default may assume those states peculiar to the particular life cycle they have been assigned.

Status	Description
IN EDITING	Changes to the document are in progress and are yet to be completed.
FOR REVIEW	Changes have been made to the document and are being reviewed.
FOR APPROVAL	The document has been reviewed and is yet to be approved.
REJECTED	The document has not been granted approval. Successive attempts to receive approval for the document at hand can be made, following its modification and revision.
APPROVED	The document has been approved.
CANCELLED	The document has been cancelled (i.e. it will no longer be considered for approval).

12.3 Documents without a life cycle

Documents without a life cycle can be edited with success, provided that the Check-Out/Check-In procedure be followed.

12.3.1 Check-Out/Check-In procedure

Only documents that do not have a life cycle require that the Check-Out/Check-In procedure be performed in order to permit their editing and versioning. The terms "Checked-Out" and "Checked-In" reflect a particular meta-status, peculiar solely to such documents.

The Check-Out/Check-In procedure maintains a full audit trail of a document's previous versions.

The exact order of the operations to be performed when editing a document without a life cycle is given below:

1. Check-Out
2. Modification
3. Save
4. Check-In.

An attempt to view or edit a checked-in document extracts a local Read-Only copy from the DB. Any changes made are "temporary" and will concern only this local copy, not the “original” document stored in the DB.

12.3.2 Document accessibility

The following table illustrates the relationship that exists between Check-Out/Check-In and document accessibility.

Meta-Status	Description	Accessibility
CHECKED-OUT	Changes are being made to the document, but are not yet effective.	Limited to the user who checked out the document. The Check Out option will be disabled in those menus where present: to find out the name of the user who has the document currently checked-out and the URL of the "working” document, see the document's Properties dialog box (Versions tab).
CHECKED-IN	The changes made to the document during the checked-out phase become effective, and a new version of the document is generated.	Can be viewed to all users who possess the appropriate Access Rights. Any attempts to edit the document will be unsuccessful, as changes must be made only to checked-out documents, which must then be checked-in, in order to become effective.