PAYROLL ENGINE

Payroll Engine Whitepaper

PayrollEngine.org
Beta Release 24.07.2023

1 Einführung

Dieses Dokument beschreibt die Funktionsweise der Payroll Engine und die zugrunde liegenden Konzepte. In den Kapiteln 1 bis 4 wird die Payroll Engine aus betriebswirtschaftlicher Sicht beschrieben. Die Kapitel 5 bis 8 beschreiben die technischen Aspekte der Lösung.

Mit der Payroll Engine können Lohndaten von Unternehmen berechnet werden. Die Engine unterscheidet sich von herkömmlichen Payroll-Systemen in folgenden Punkten:

Fallgetrieben	Datenänderungen erfolgen fallbezogen und werden der Zeitachse zugeordnet. Dadurch sind die Geschäftsdaten zu jedem Zeitpunkt gültig. Automatisierte Lohnläufe inklusive Retro- und Forecast-Szenarien sind jederzeit möglich.
Globalisierung	Pro Mandant können die Geschäftsdaten mit den jeweiligen landes- und firmenspezifischen Gesetzen und lokalen Vorschriften abgerechnet werden.
Regelwerke	Die Lohnanwendung wird in Regelwerken bestimmt, welche als Komponenten schichtweise aufeinander aufbauen. Es bestehen Regelwerke für Länder, Branchen und Unternehmen, wie z.B. allgemeinverbindliche Gesamtarbeitsverträge oder Regelwerke für Pensionskassen.
Kundenskalierung	Kundenanpassungen werden ebenfalls als Regelwerk implementiert, so dass skalierbare Lösungen für kleine bis große Unternehmen möglich sind.
Automatisierung	Die Engine verfügt über ein konfigurierbares und erweiterbares Daten- und Verarbeitungsmodell an, das über die REST-API definiert wird. Das Laufzeitverhalten der Lohnanwendung wird über die Scripting-API gesteuert.

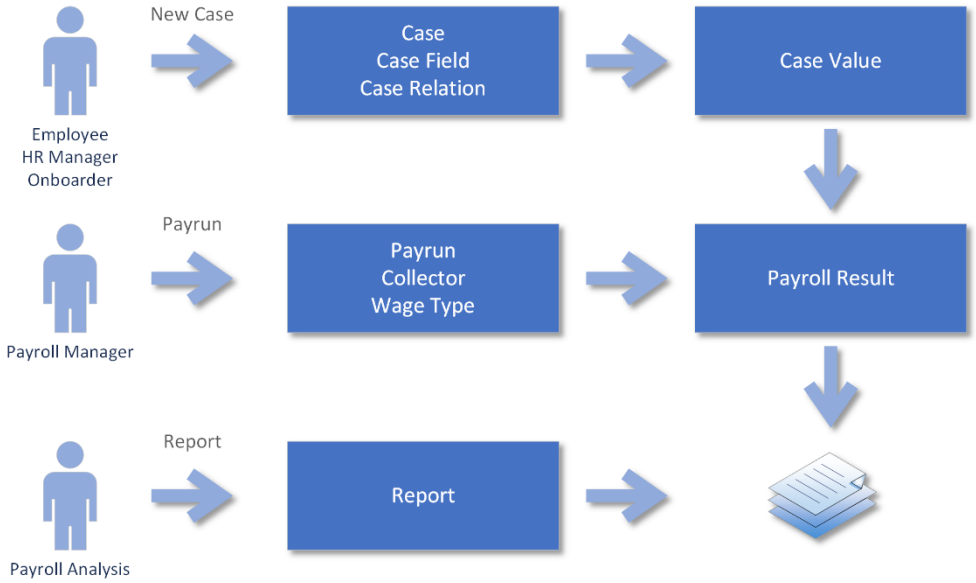
2 Payroll-Modell

2.1 Zentrale Anwendungsfälle

Die zentralen Payroll-Anwendungsfälle decken folgende Bereiche ab:

- Fälle Management: Änderung von Unternehmens- und Mitarbeiterdaten
- Lohnlauf: Zyklische Berechnung der Mitarbeiterlöhne
- Report: Auswertung der Lohndaten

Die folgende Abbildung zeigt die Payroll Objekte der zentralen Anwendungsfälle:

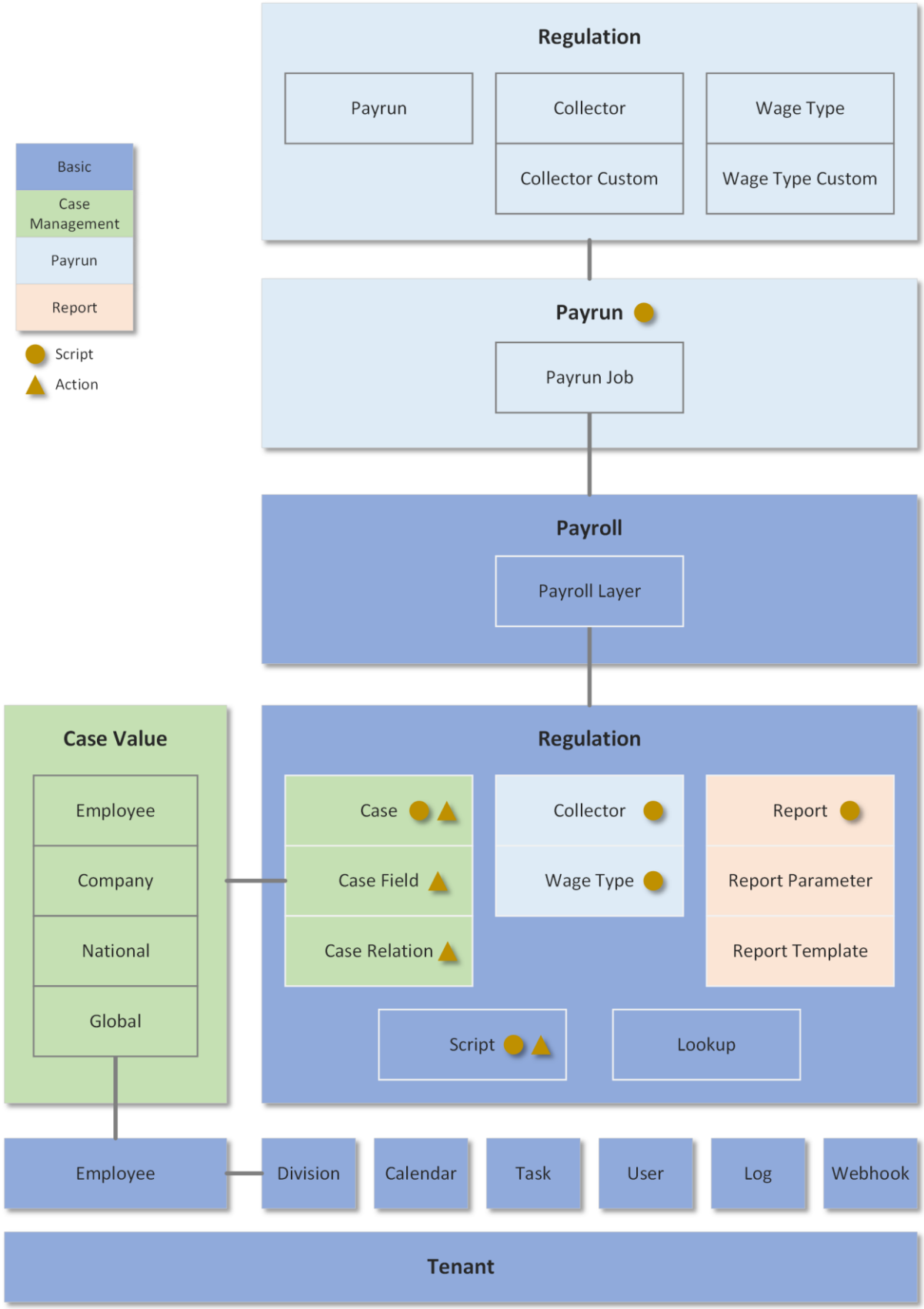


2.2 Modellübersicht

Das Payroll-Modell enthält Objekte für die folgenden Funktionsbereiche:

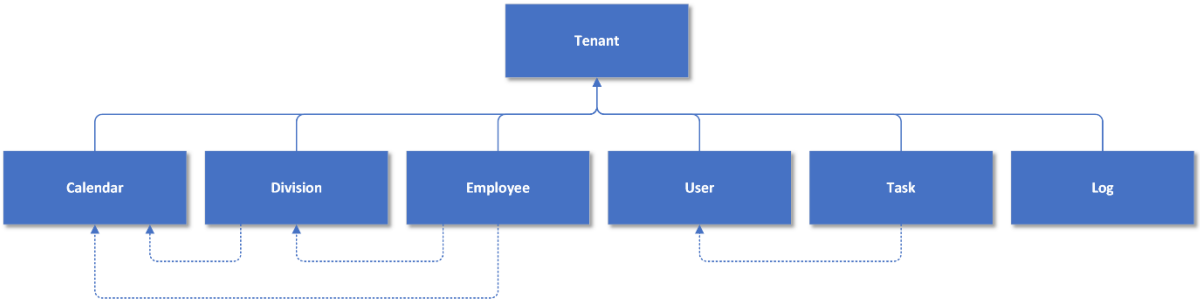
- Basis und Infrastruktur Objekte (*Basic*)
- Objekte zur Erfassung von Geschäftsdaten (*Case Management)*
- Objekte für die Verarbeitung der Lohndaten (*Payrun*)
- Objekte zur Transformation der Ausgabedaten (*Report*)

Die folgende Übersicht zeigt die statische Topologie sowie die Steuerung des Laufzeitverhaltens mit Scripts und Actions:



2.3 Mandant

Der mandantenbasierte Payroll Service bietet folgende Objekte zur Definition des Mandanten mit seinen Organisationen, Mitarbeitern und Benutzern:

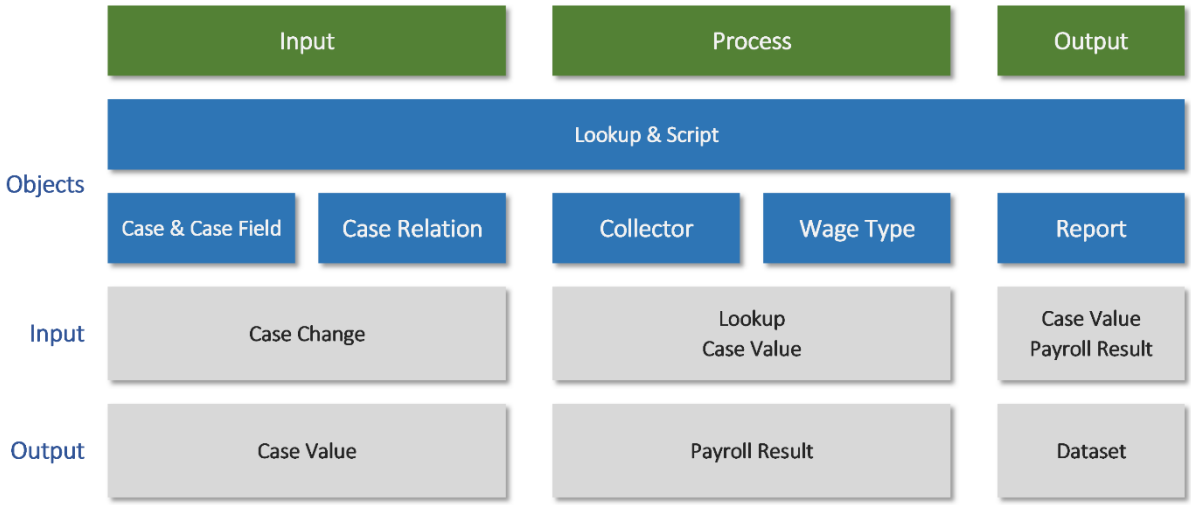


Tenant	Beinhaltet alle Mandatsdaten. Auf dieses Objekte verweisen alle Mandatsdaten.
Calendar	Die Lohnkalender des Mandanten.
Division	Unterteilt das Unternehmen in Unternehmensbereich. Jeder Mandant besteht aus mindestens einem Unternehmensbereich.
Employee	Mitarbeiter, die einem oder mehreren Unternehmensbereichen zugeordnet sind.
User	Benutzer der Systems.
Task	Benutzertasks.
Log	Mandantenspezifische Logeinträge.

Mit Ausnahme der geteilten Regelwerke (siehe Geteilte Regelwerke), sind alle im Folgenden beschriebenen Domänenobjekte dem Mandanten zugeordnet.

2.4 Regelwerk

Ein Regelwerk umfasst die Definition des Lohns mit den Elementen der Lohndateneingabe, -verarbeitung und -ausgabe:

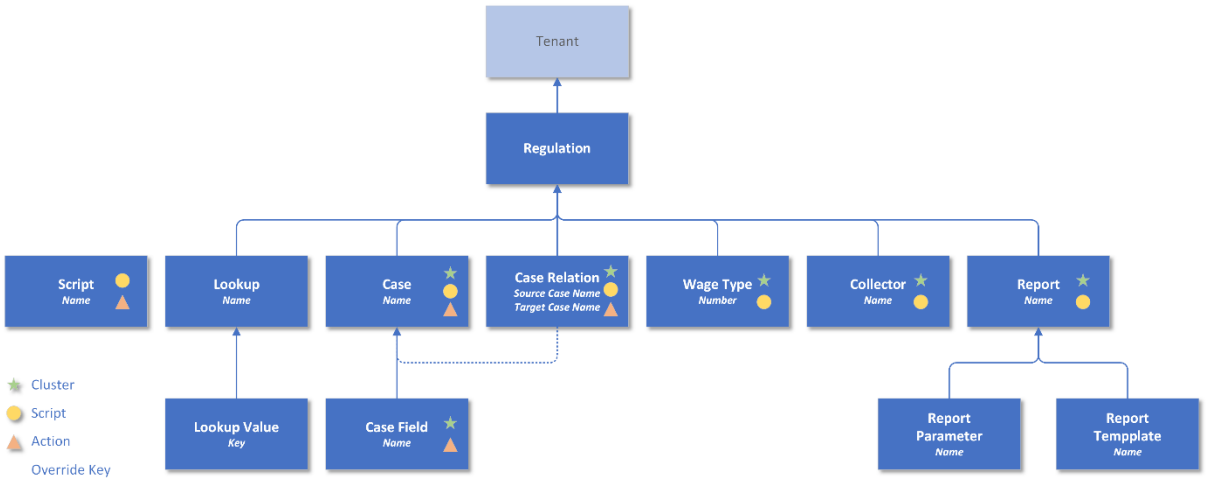


Die Regelwerke behandelt folgende Themenbereiche:

- Definition des Datenmodells und der Geschäftsregeln, siehe Fall mit Felder
- Berechnung der Lohndaten, siehe Lohnlauf
- Auswertung der Firmen- und Mitarbeiterdaten, siehe Report
- Testen der Firmen- und Mitarbeiterdaten, siehe Payroll Tests
- Bestehende Regelwerke erweitern, siehe Vererbung von Regelwerken
- Software zwischen Mandanten teilen, siehe Geteilte Regelwerke
- Software bereitstellen, siehe Regelwerke entwickeln und bereitstellen

2.4.1 Regelwerksobjekte

Das Regelwerk umfasst folgende Objekte:



Regulation	Beinhaltet die Daten eines Regelwerks.
Script	Gemeinsamer Code für die Funktionen innerhalb der Regelung.
Lookup **Lookup Value**	Selektionsdaten für Anwendungen (z.B. Geschlecht) und Wertetabellen für Scripting-Funktionen (z.B. Steuertabellen).
Case	Lohn Anwendungsfall in den Bereichen Global, National, Unternehmen und Mitarbeiter.
Case Field	Feld, das den Wert des Falles beschreibt. Die Mitarbeiterdaten können für einen oder alle Unternehmensbereiche gelten.
Case Relation	Regelt die Abhängigkeit zwischen zwei Fällen.
Wage Type	Bestimmt die Lohnart und ihre Berechnung. Die Lohnarten werden in der Reihenfolge der Lohnartennummer mit einem Dezimalwert (z.B. 1000) berechnet. Durch Nachkommastellen in der Lohnartennummer (z.B. 1000.1) können untergeordnete Lohnarten in die Verarbeitungsreihenfolge einbezogen werden. Die Lohnart bestimmt, welche Kollektoren bzw. Kollektorgruppen relevant sind.
Collector	Aggregiert die Lohndaten während des Lohnlaufs für z.B. für die Lohnbasen. Die Art des Kollektors wird durch den Aggregationstyp gesteuert z.B. Summierung, Minimum, Maximum, Zählen usw.. Kollektoren können einzeln oder in Gruppen ausgeführt werden.

Report	Definition des Reports.
Report Parameter	Parameter des Reports.
Report Template	Berichtsvorlage in einer Sprache.

Durch Überlagerungen kann ein Regelwerksobjekt durch ein übergeordnetes Branchen- oder Kundenregelwerk übersteuert werden (siehe Payroll). Die Identifikation des zu übersteuernden Objektes erfolgt über den Überlagerungsschlüssel, z.B. den Fallnamen.

Clusterobjekte bieten einen erweiterten Tagging-Mechanismus, um gleichartige Objekte nach Anwendungskriterien zu berücksichtigen oder auszuschließen (siehe Payroll Clusters).

Für jedes Regelwerksobjekt existiert ein Audit Trail (siehe Audit Trail) mit entsprechenden REST-Endpunkten. Änderungen an einem Regelwerk (z.B. jährlicher Update) werden in Versionen verwaltet, wobei jede Version ab einer bestimmtem Abrechnungsperiode gültig ist. Dies gewährleistet die korrekten Zeitdaten der Lohnläufe bei Rückrechnung und Forecast.

2.4.2 Anwendungsszenarien

Folgende Einsatzgebiete sind für die Payroll- Regelwerke möglich:

- Entwicklung kundenspezifischer Regelwerke im Onboarding
- Dienstleister bietet kommerzielle Regelwerke an
- Branchenlösungen wie z.B. Verbandslösungen
- Länderspezifische und gesetzliche Abrechnungen
- Unternehmens-Regelwerke, lokal oder transnational
- Regelwerke für Tests und Weiterentwicklungen

2.4.3 Vererbung von Regelwerken

Die Regelwerke basieren auf dem Prinzip der Vererbung, d.h. ein Objekt eines Regelwerkes kann durch ein Objekt eines Regelwerkes einer höheren Ebene übersteuert werden. Die zu überschreibenden Objekte werden durch den Überschreibschlüssel identifiziert:

- *Case* Name (Schlüssel)
- *Case Field* Name
- *Case Relation* Fallnamen von Quelle und Ziel
- *Collector* Name
- *Wage Type* Lohnartennummer
- *Report* Name
- *Report Parameter* Name
- *Report Template* Name
- *Lookup* Name
- *Lookup Value* Schlüssel

2.4.4 Fall mit Felder

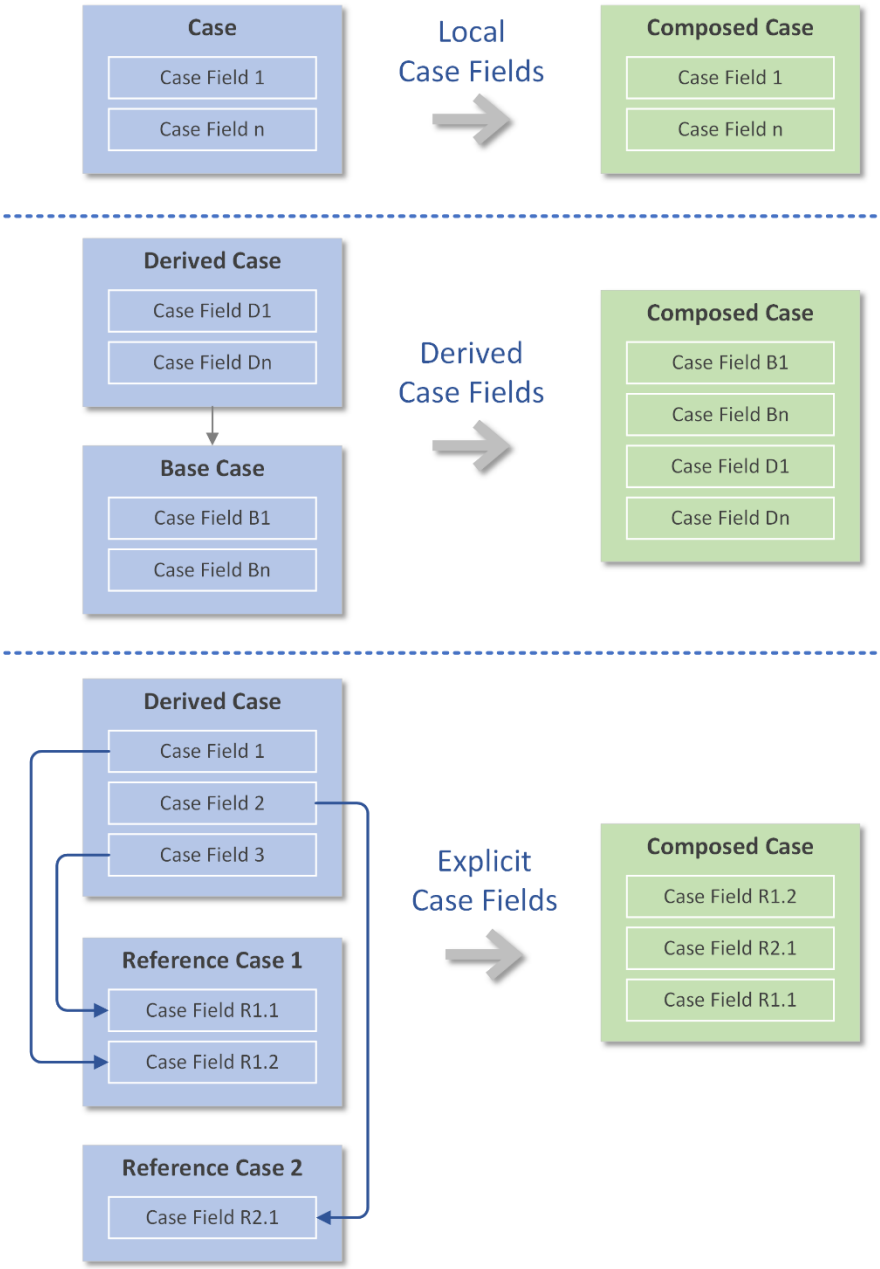
Das Objekt *Case* dient der Erfassung von Falldaten. Das Objekt *Case Field* repräsentiert ein Eingabefeld im Fall. Das Objekt *Case Relation* definiert die Beziehung (Verfügbarkeit und Wertübergabe) zwischen zwei Fällen.

Bei der Erfassung des Fallwertes wird dessen Gültigkeitszeitraum mit dem Datumsbereich (von/bis) festgelegt. Dies ermöglicht die Abbildung komplexer Geschäftsfälle, wie z.B. die Stornierung früherer Mutationen oder die Terminierung von Lohnanpassungen (siehe Zeittypen). Diese Eigenschaft unterscheidet die Payroll Engine wesentlich von herkömmlichen Softwarelösungen, die in der Regel bestehende Daten überschreiben.

Die Engine bietet verschiedene Ansätze für die Zusammenstellung von Fällen und deren Felder:

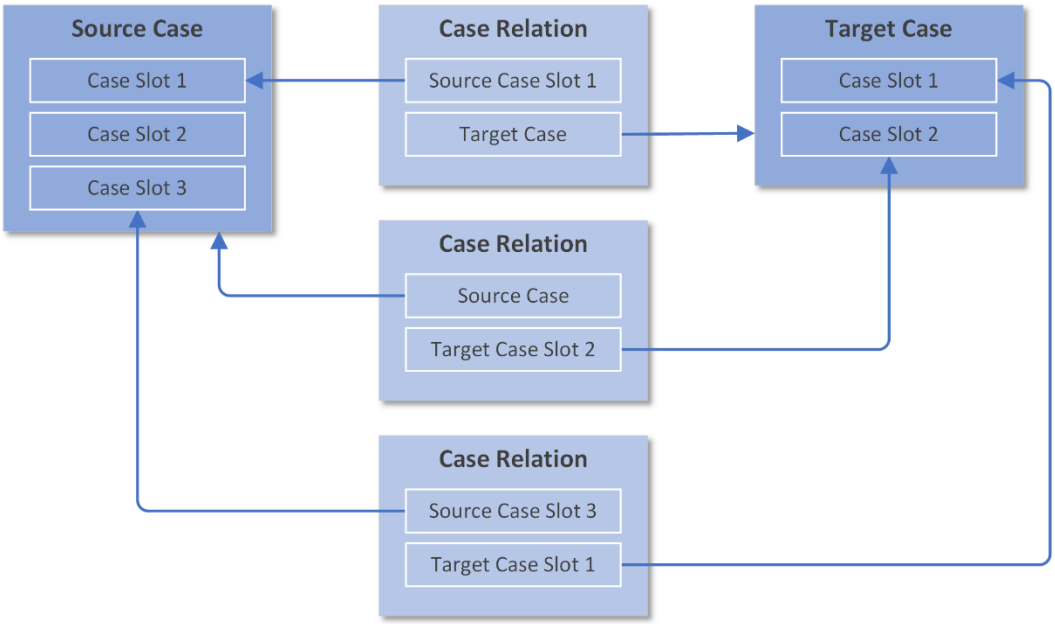
- Deklaration der Fallfeldes im Fall als integraler Bestandteil
- Übernahme von Fallfeldern aus einem anderen Fall
- Einbinden von Fallfeldern aus verschiedenen Fällen

Die folgende Übersicht zeigt mögliche Fallkonstellationen, die auch kombiniert werden können:



2.4.5 Fall Slots

Um mehrere Datensätze zu einem Fall (z.B. Kind oder Adresse) speichern zu können, wird jede Variante einem Fall-Slot mit einem eindeutigen Namen zugeordnet.



2.4.6 Lookups

Das Lookup enthält Selektionsdaten eines Regelwerks für

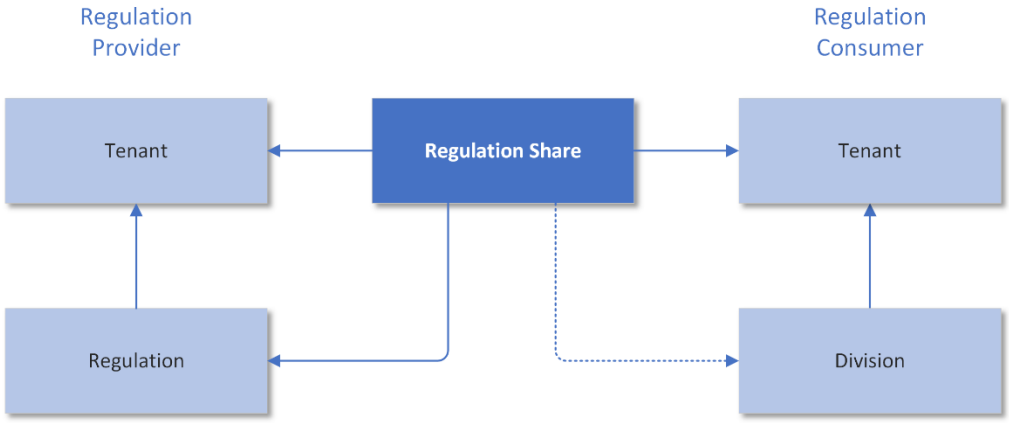
- fremdbestimmte Daten von externen Stellen, z.B. einer Steuertabelle
- unveränderliche Selektionswerte, die sich während einer Abrechnungsperiode nicht ändern
- Werte, die sich zyklisch oder periodisch ändern können

Ein Lookup besteht aus einer Werteliste (*Lookup Value*), die sich aus einem Schlüssel und einem Wert zusammensetzt. Der Schlüssel ist ein Text, der innerhalb eines Lookups eindeutig ist. Der Lookup-Wert wird durch eine JSON-Struktur beschrieben. Der Benutzer ist dafür verantwortlich, dass die JSON-Struktur für alle Werte eines Lookups strukturell identisch ist.

Neben der Suche nach einem Lookup Value über den Schlüssel kann der Lookup Value auch über eine Wertigkeit (Dezimalzahl) gesucht werden. Dies kann z.B. für einkommensabhängige Steuertabellen verwendet werden.

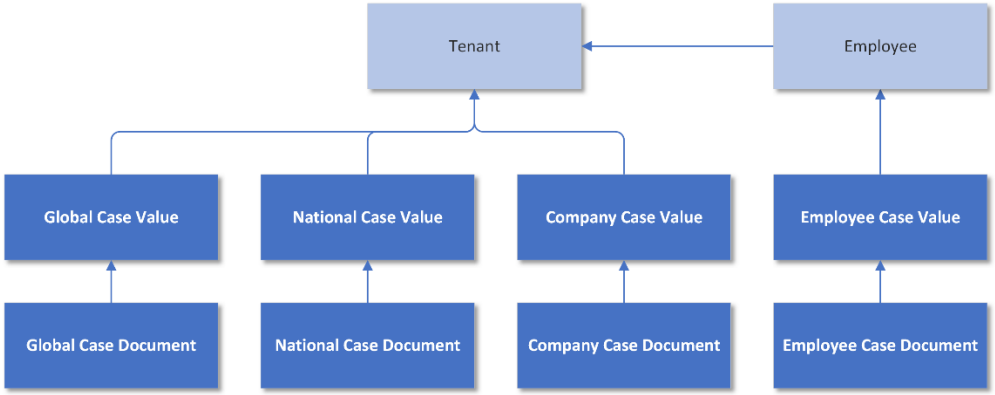
2.5 Geteilte Regelwerke

Die Engine bietet die Möglichkeit, ein Regelwerk für mehrere Mandanten freizugeben. Dies ermöglicht dem Payroll Provider die zentrale Administration von Länder- und Branchenregelwerken. Über Berechtigungen wird festgelegt, welches Regelwerk für welchen Mandanten (optional Unternehmensbereich) verfügbar ist:



2.6 Falldaten

Für jedes Feld eines Falles (*Case Field*) kann ein Wert (*Case Value*) durch eine Falländerung (*Case Change*) erfasst werden. Die Falldaten werden physisch getrennt nach Nationalität, Unternehmen und Mitarbeiter gespeichert:



Global Case Value	Die Globalen Daten.
Global Case Document	Die Globalen Dokumente.
National Case Value	Die Nationalen Daten.
National Case Document	Die Nationalen Dokumente.
Company Case Value	Die Unternehmensdaten.
Company Case Document	Die Unternehmensdokumente.
Employee Case Value	Die Mitarbeiterdaten.
Employee Case Document	Die Mitarbeiterdokumente.

Ein Fallwert steht wahlweise für alle oder für einen Unternehmensbereich zur Verfügung. Die Definition erfolgt im Datenfeld (*Case Field*) des Regelwerks.

Die Falldaten sind unveränderbar und werden durch die Falländerungen ergänzt. Anhand dieser Änderungshistorie können die Daten einer Lohnperiode bestimmt werden.

Der Wert eines Falles (Case Value) kann auf einen Gültigkeitszeitraum (Beginn/Ende) beschränkt werden (siehe Zeittypen). Der Lohnlauf berücksichtigt den Wert nur, wenn sein Zeitraum mit der Lohnperiode übereinstimmt.

Die Falltypen können in den Regelwerken wie folgt verwendet werden:

Regelwerk	Global	National	Unternehmen	Mitarbeiter
Mandant	✓	✓	✓	✓
Business	✓	✓	✓	✓
National	✓	✓		
Global	✓			

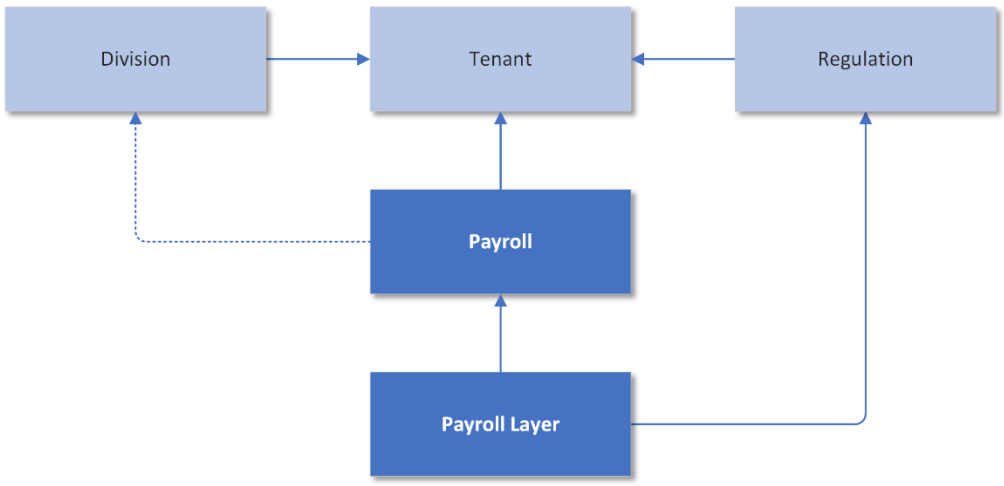
2.6.1 Falldaten stornieren

Ein Fall der Stornierungen zulässt, steuert das Rücksetzverhalten für nummerische Fallwerte über den Stornotyp:

- Übernahme vorheriger Wert
- Wert auf Default initialisieren
- Wert invertieren
- Wert beibehalten

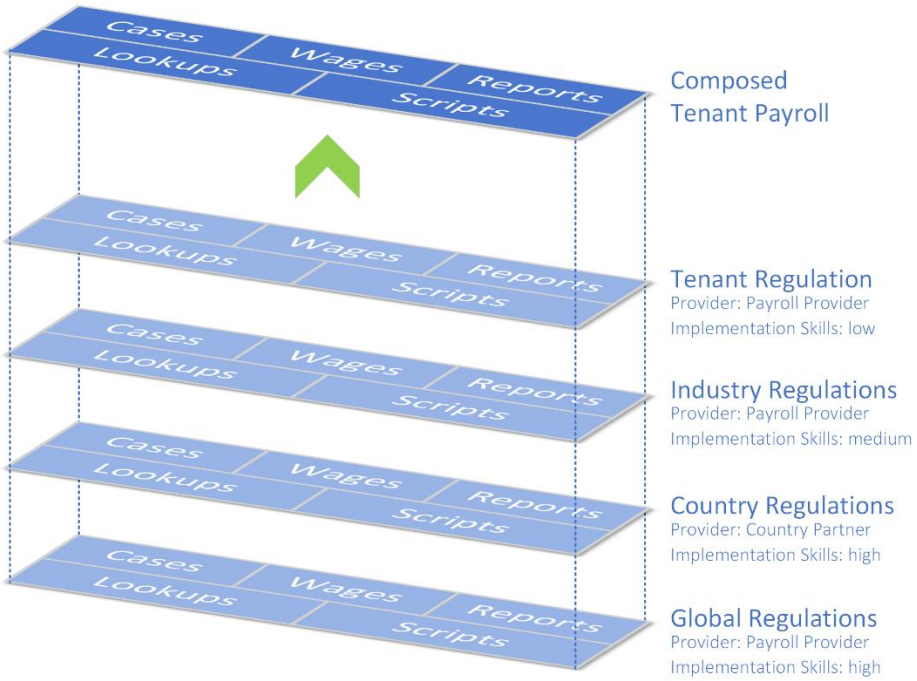
2.7 Payroll

Die Payroll verbindet mehrere Regelwerke über ein Schichtenmodell zu einem virtuellen Lohnmodell. Folgende Objekte sind daran beteiligt:



Payroll	Fasst mehrere Regelwerksschichten zu einem Payroll zusammen und ist einer Unternehmenseinheit zugeordnet.
Payroll Layer	Bestimmt die Prioritäten und Reihenfolge der Regelwerke.

Die Payroll bietet eine konsolidierte Sicht auf die Regelwerke:



2.7.1 Payroll Schichten

Die Auswertungsreihenfolge der Regelwerke wird im *Payroll Layer* über den Level (1. Sortierkriterium) und die Priorität (2. Sortierkriterium) festgelegt. Die Anzahl der Schichten ist unbegrenzt. Das folgende Beispiel zeigt drei Regelwerksschichten, die zu einem dynamischen Payroll verknüpft sind:

Composed Regulation	Address	Address.Zip [2.2] Address.Line [2.1] Address.City [1.1] Address.Street [1.1]	Address > Partner [2.1]	1500 [2.2] 1000 [2.1]	Tax C [2.2] Tax A [1.1]

Payroll Layer		Regulation				
Level	Priority	Case	Case Field	Case Relation	Wage Type	Collector
2	2	Address	Address.Zip	Priority	1500	Tax C
2	1	Address	Address.Line	Address > Partner	1000	Priority
1	1	Address	Adress.Zip Address.City Address.Street	Address > Partner	1000	Tax A

2.7.2 Objektvererbung

Anhang des Überlagerungsschlüssels können Objekte aus unteren Schichten übersteuert werden. Dabei werden die Werte aller Schichten dynamisch zu einem Objekt zusammengefasst:

- Lokalisierungen
- Attribute

- Clusters
- Case Slots, Lookups und Actions
- Textfelder

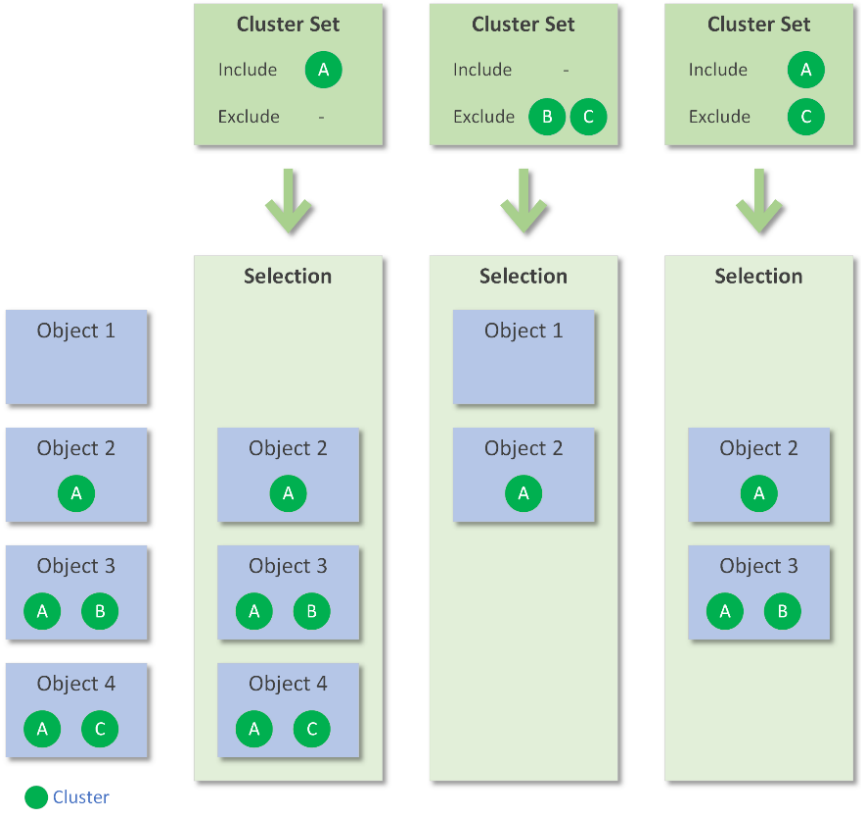
2.7.3 Payroll Clusters

Mittels Cluster können Regelwerksobjekte nach beliebigen Kriterien gruppiert werden. Dies kann verwendet werden, um:

- API-Abfragen zu filtern z.B. zum Gruppieren von Fällen
- durch Reduzieren von Falldaten den Lohnlauf optimieren
- die Payroll Resultate einzuschränken

Jedem Clusterobjekt können mehrere Clusternamen (analog Tags) zugeordnet werden. Die Abfrage der Objekte erfolgt über das Cluster Set, das die einzubeziehenden (Whitelist) bzw. auszuschließenden (Blacklist) Cluster enthält. Durch die Kombination von Whitelist-Clustern mit Blacklist-Clustern, können die berücksichtigen Clusterobjekte weiter eingeschränkt werden.

Die folgende Abbildung zeigt verschiedene Szenarien für die Auswahl von Clusterobjekten:



Cluster Sets sind im Objekt *Payroll* definiert und können im Lohnlauf verendet werden. Es gibt folgende Cluster Sets:

- *Case Cluster Set*: verfügbare Eingabefälle
- *Case Field Cluster Set*: verfügbare Falldaten im Lohnlauf
- *Collector Cluster Set*: verfügbare Kollektoren im Lohnlauf
- *Collector Retro Cluster Set*: verfügbare Kollektoren im Rückrechnungs-Lohnlauf
- *Wage Type Cluster Set*: verfügbare Lohnarten im Lohnlauf
- *Wage Type Retro Cluster Set*: verfügbare Lohnarten im Rückrechnungs-Lohnlauf
- *Wage Type Period Cluster Set*: Periodenresultate von Falldaten erzeugen
- *Case Value Cluster Set*: verfügbare Falldaten im Lohnlauf für Zusatzresultate

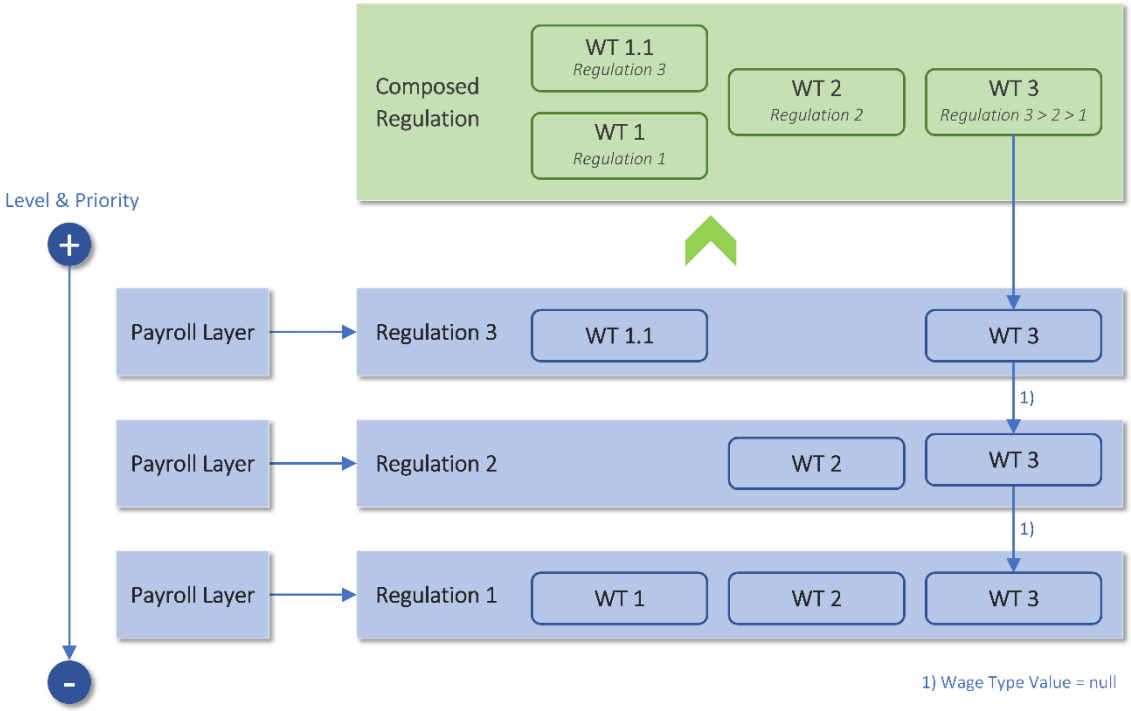
2.7.4 Payroll Scripting

Die Scripting API ermöglicht die Steuerung des Laufzeitverhaltens für spezifische Payroll-Objekte (siehe Log für Lohnläufe (siehe Lohnlauf-Logs)

- Log für Reports (siehe Report-Logs)

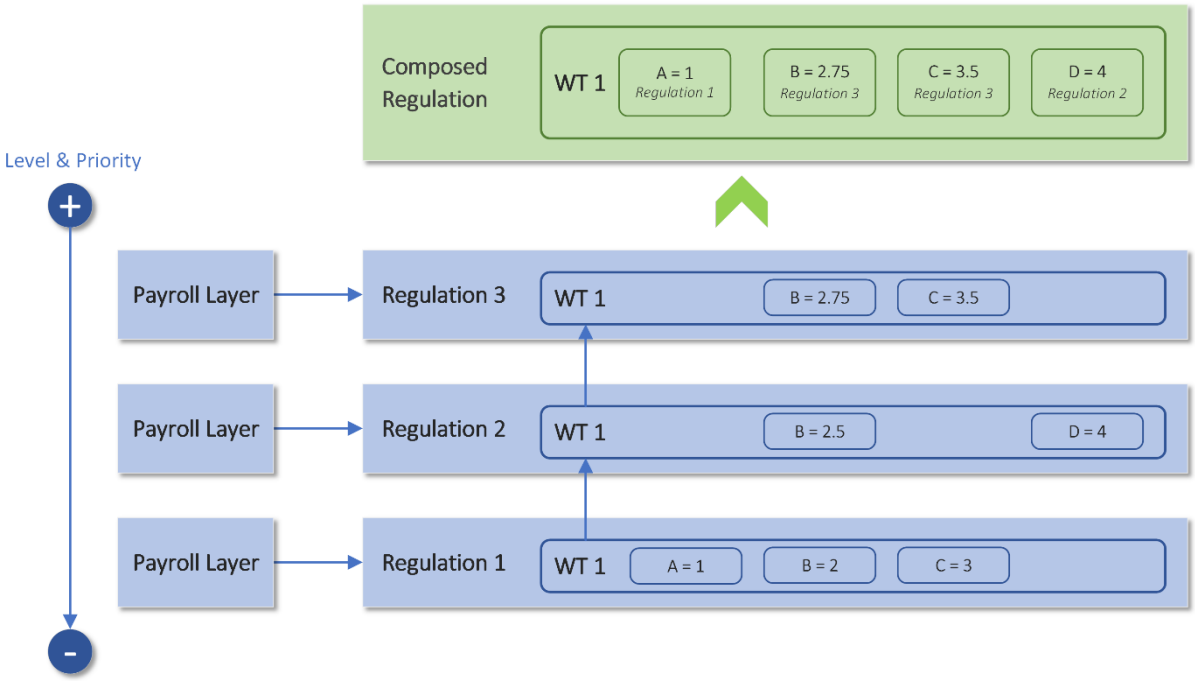
Scripting API). Die Auswertung der Skripte erfolgt über die Auswertungsreihenfolge im Payroll Layer (siehe Payroll Schichten).

Das folgende Beispiel zeigt die Berechnung der Lohnarten:



Durch die Rückgabe eines undefinierten Rückgabewertes (*null*) einer Scripting-Funktion, im obigen Beispiel WT2, wird die Verarbeitung an das zugrundeliegende Regelwerk delegiert.

Im Gegensatz dazu werden Objektattribute in umgekehrter Richtung ausgewertet, so dass der Wert durch ein übergeordnetes Regelwerk überschrieben werden kann:



2.8 Lohndatenberechnung

2.8.1 Lohnarten

Die Berechnung der Löhne der Mitarbeiter erfolgt mit Hilfe von Lohnarten, die in der Reihenfolge ihrer Lohnartennummer berechnet werden. Jeder Lohnart können ein oder mehrere Kollektoren zugeordnet werden, die die fortlaufenden Lohnbasen berechnen. Die Anzahl der Lohnarten ist nicht begrenzt und ergibt sich aus der Summe der Payroll-Regelwerke.

2.8.2 Kollektoren

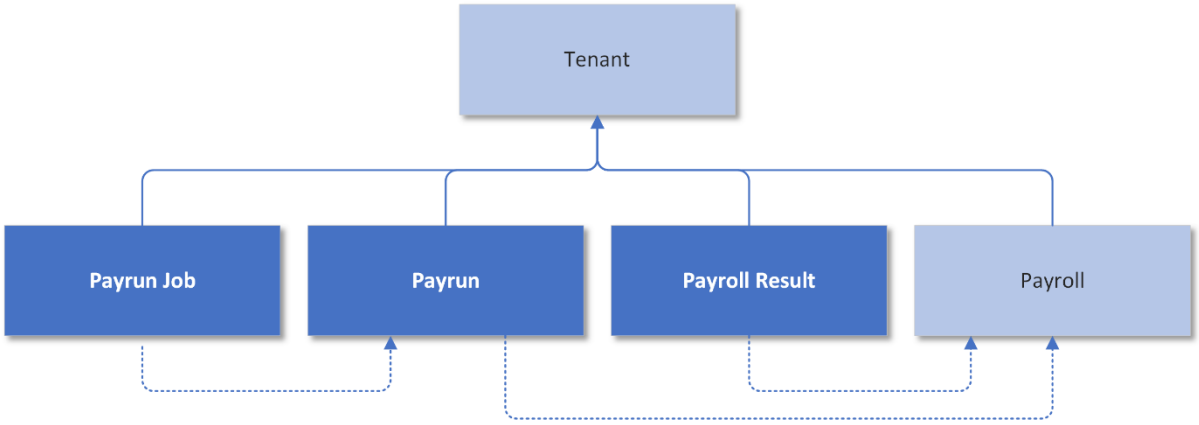
Kollektoren sammeln die Werte der Lohnarten zusammen und bieten verschiedene vordefinierte Operatoren:

- *Sum* (Default)
- *Min*
- *Max*
- *Average*
- *Count*

Zusätzlich zu den vordefinierten Aggregationstypen kann der Kollektorwert mit der Scripting-Funktion *CollectorApply* individuell bestimmt werden.

2.9 Lohnlauf

Die Berechnung der Lohndaten erfolgt mit folgenden Objekten:



Payrun Job	Die Ausführung des Lohnlaufes.
Payrun	Die Definition des Lohnlaufes.
Payroll Result	Die Ergebnisse des Lohnlaufes.

Die Lohnlauf Sequenz ist im Kapitel Regelwerk Scripting beschrieben.

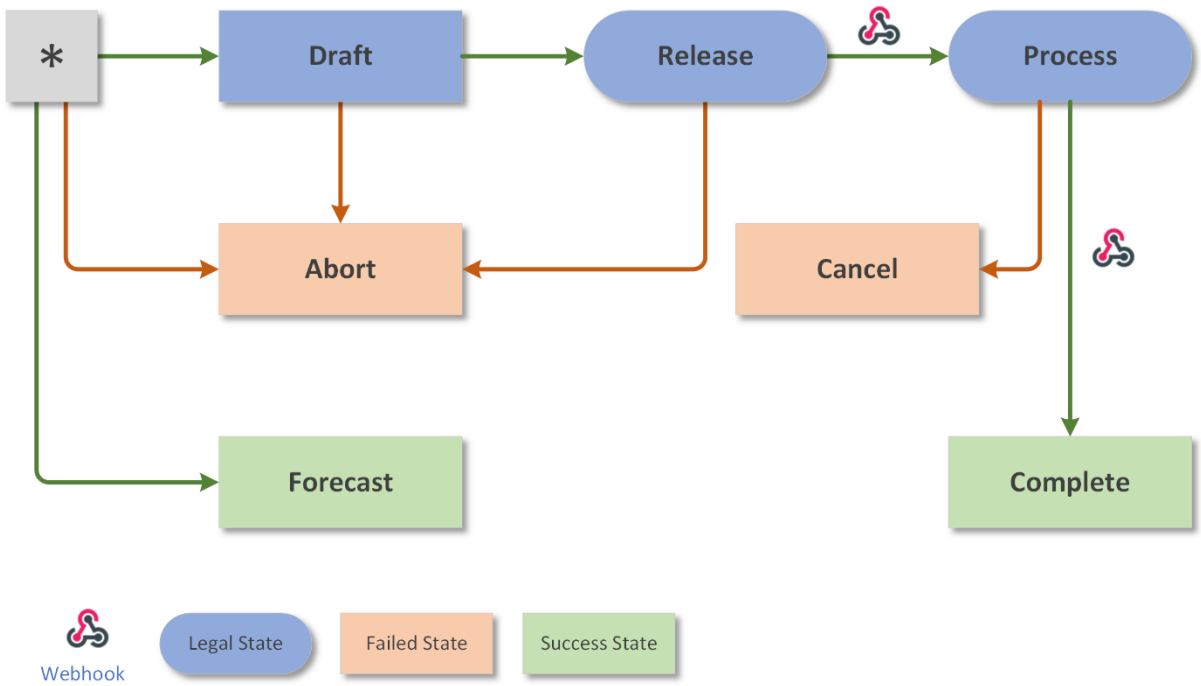
2.9.1 Lohnlauf Job

Der Lohnlauf Job (*Payrun Job*) startet den Lohnlauf (*Payrun*) für eine Lohnperiode und speichert die Ergebnisse im Ergebnis (*Payroll Result*). Die zugrundeliegende Payroll bestimmt anhand des Unternehmensbereiches, ob der Mitarbeiter zu berücksichtigen ist.

Der Lohnlauf Job bestimmt, für welchen Zweck die Ausführung erfolgt:

- Legal: Gesetzliche Lohnmeldung
- Forecast: Analyse der Lohndaten für Prognosen, Fallszenarien etc.

Die Steuerung des Lohnlauf-Jobs erfolgt über den Jobstatus:



Jobstatus	Typ	Beschreibung	Webhook
*		Neuer Lohnlauf-Job	
Draft	Working	Gesetzlicher Lohnlauf-Job zur Vorschau	
Release	Working	Gesetzlicher Lohnlauf-Job freigegeben zur Verarbeitung	
Process	Working	Gesetzlicher Lohnlauf-Job in Verarbeitung	*PayrunJobProcess*
Complete	Final	Gesetzlicher Lohnlauf-Job erfolgreich verarbeitet	*PayrunJobFinish*
Forecast	Final	Forecast Lohnlauf-Job	
Abort	Final	Gesetzlicher Lohnlauf-Job vor Freigabe abgebrochen	
Cancel	Final	Gesetzlicher Lohnlauf-Job fehlerhaft verarbeitet	*PayrunJobFinish*

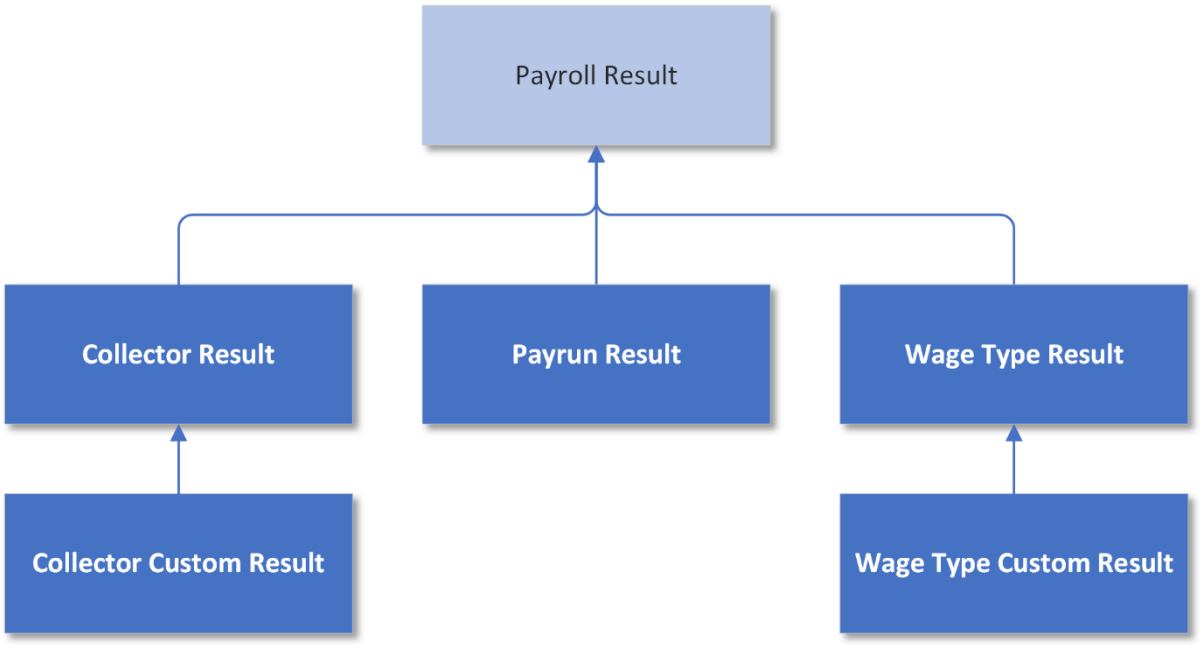
Für die gesetzlichen Lohnläufe kann es pro Lohnlauftyp (*Payrun*) und Lohnperiode nur einen Job im Status *Draft* geben. In den Status *Release* und *Process* sind mehrere Jobs mit demselben Status möglich. Forecast Lohnläufe können beliebig oft ausgeführt werden.

2.9.2 Lohnlauf Neustart

Während des Lohnlaufs werden alle Lohnarten in der Reihenfolge der Lohnartennummern verarbeitet. In Sonderfällen kann der Lohnlauf für einen Mitarbeiter erneut gestartet werden. Jeder Lauf wird in der Lohnart durch den Laufzähler gekennzeichnet. Mit Hilfe der Laufzeitwerte (siehe Lohnlauf Scripting) können Daten zwischen den Läufen ausgetauscht werden.

2.9.3 Payroll Resultate

Die Ergebnisse des Lohnlaufes werden in verschiedenen Objekten gespeichert:



CollectorResult	Das Ergebnis (Dezimalwert) des Kollektors.
Collector Custom Result	Benutzerdefiniertes Kollektor Ergebnis (Dezimalwert).
PayrunResult	Lohnlaufspezifisches Ergebnis (auch Fallwerte).
Wage Type Result	Das Ergebnis der Lohnart (Dezimalwert) inkl. benutzerdefinierten Attributen.
Wage Type CustomResult	Benutzerdefinierter Lohnergebnis (Dezimalwert).

In der Payroll wird mit Cluster Sets (siehe Payroll Clusters) gesteuert, welche Ergebnisse erzeugt werden sollen.

2.10 Spezielle Lohnläufe

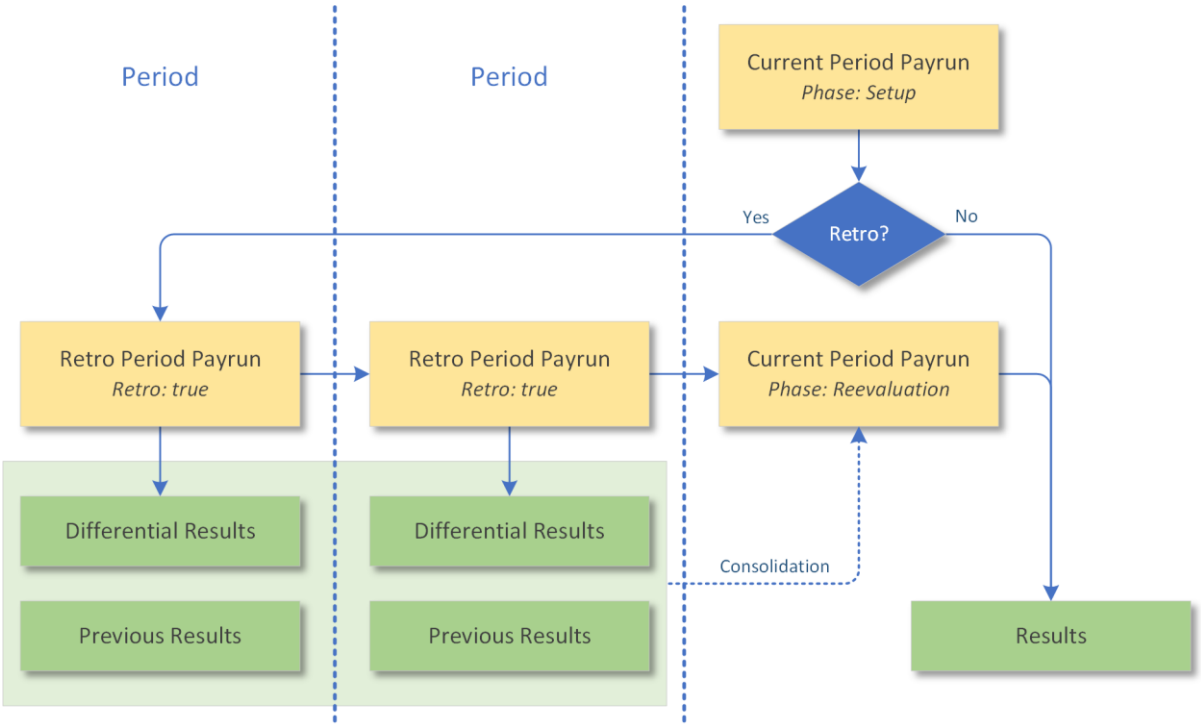
2.10.1 Inkrementeller Lohnlauf

Wenn innerhalb einer Periode mehrere Lohnläufe durchgeführt werden, speichert die Engine nur inkrementelle Ergebnisse, d.h. Werte, die sich seit dem letzten Lohnlauf geändert haben. Um die aktuell gültigen (konsolidierten) Ergebnisse einer Lohnperiode zu ermitteln, bietet die REST API entsprechende Endpunkte an.

2.10.2 Rückrechnung

Rückrechnungen erfolgen automatisch, wenn seit dem letzten Lohnlauf neue Mutationen aufgetreten sind, die sich auf vergangene Lohnperioden auswirken.

Der Rückrechnungsablauf:



Die Schritte der Rückrechnung sind:

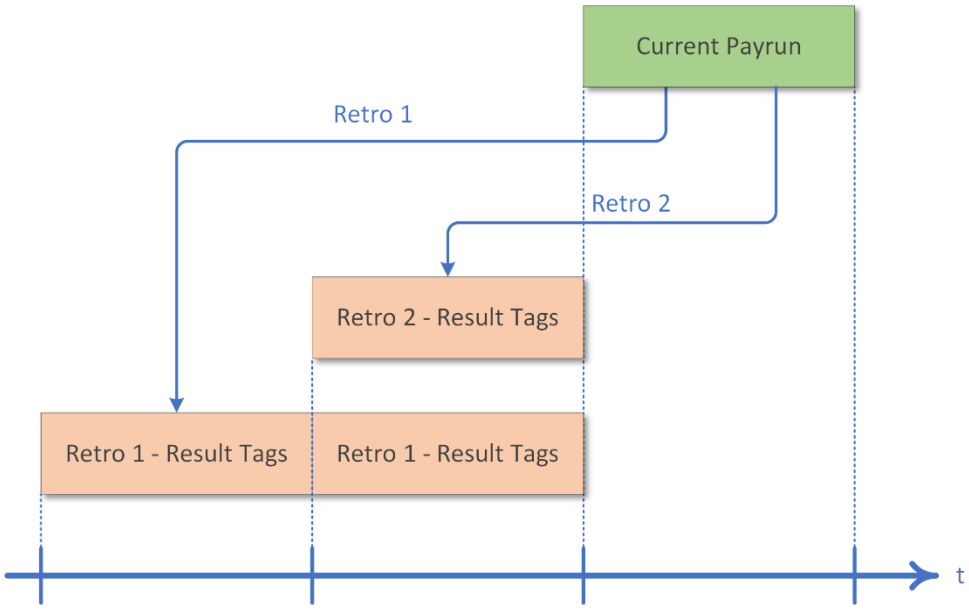
1. Berechnung der aktuellen Lohnperiode (mit Mutation in die Vorperiode)
   → Ausführungsphase *Setup*
   → Transiente Resultate
2. Berechnung aller Vorperioden beginnend bei der Mutationsperiode bis zur Vorperiode
   → Inkrementelle Resultate der Vorperioden speichern
3. Neuberechnung der aktuellen Lohnperiode
   → Ausführungsphase *Neubewertung*
   → Zugriff auf Konsolidierungsresultate unter Berücksichtigung der Retro-Resultate
   → Resultate speichern

Für Forecasts gelten die Vorläufe mit demselben Forecast-Namen. Die Anzahl der Perioden in der Rückrechnung ist unbegrenzt, es besteht jedoch die Möglichkeit, die Rückrechnung auf den Lohnzyklus zu beschränken.

2.10.3 Manuelle Rückrechnung

Neben der automatischen Rückrechnung kann die Rückrechnung auch manuell über Skripte angestossen werden. Die bei der Rückrechnung erzeugten Ergebnisse werden mit Hilfe von Payroll Result Tags gekennzeichnet. Bei der Abfrage der Abrechnungsergebnisse können die Tags als Filterkriterien verwendet werden.

Im folgenden Szenario löst der Lohnlauf der aktuellen Periode zwei Rückrechnungs-Lohnläufe aus:



2.11 Forecast

Zur Erstellung von Forecasts sind zwei Eingriffe erforderlich:

- Änderung der Falldaten für einen Forecast
- Ausführen eines Lohnlauf-Jobs für einen Forecast

Wird der Lohnlauf für einen Forecast gestartet, stehen alle Falldaten des Forecasts zur Verfügung sowie alle anderen Falldaten, die keinem Forecast zugeordnet sind. Der normale Lohnlauf hingegen ignoriert die Falldaten von Forecasts.

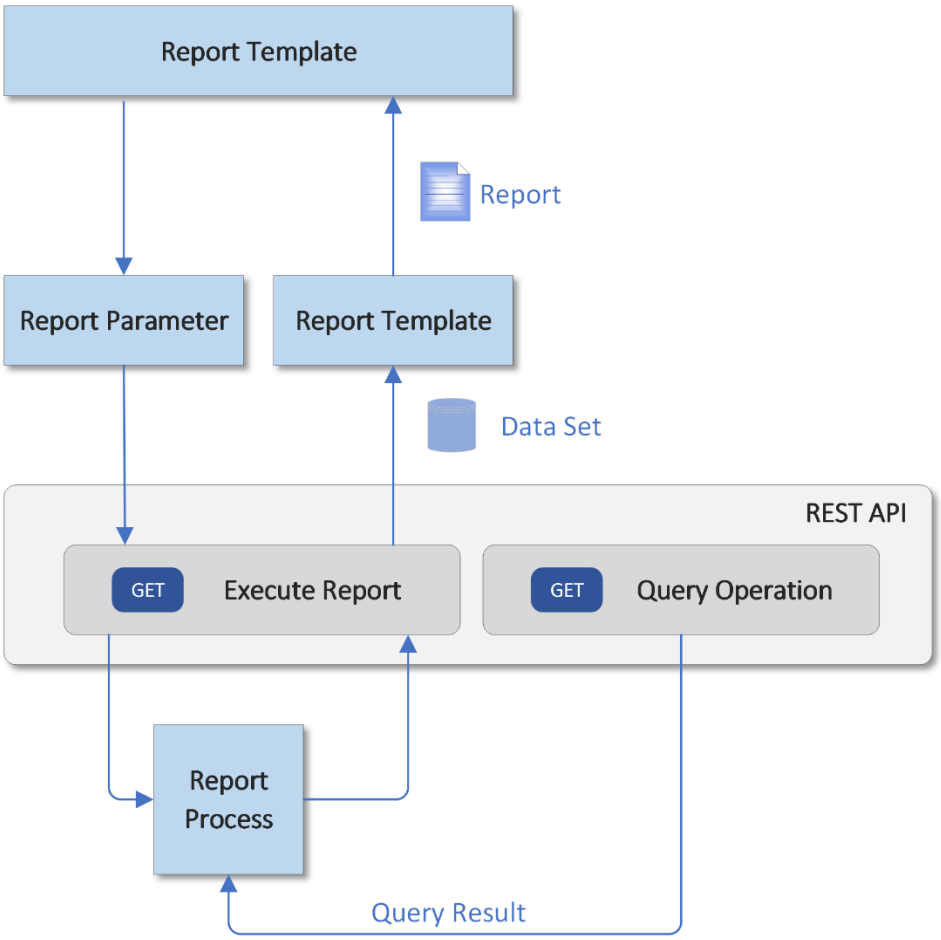
2.12 Report

Die Report-Generierung erfolgt in drei Schritten, wobei jeder Schritt optional ist:

1. Report Aufbereitung: *Report Build*
2. Reportabfragen: *Report Start*
3. Auf- oder Nachbearbeitung der Abfragedaten: *Report End*

Die Payroll API berechnet die Reportdaten in einem Dataset (Tabellen und Relationen), das vom Client in das Zieldokument transformiert wird. Zur Ermittlung der Reportdaten dienen alle GET-Endpunkte der REST-API als Datenquelle.

Report Generierung:

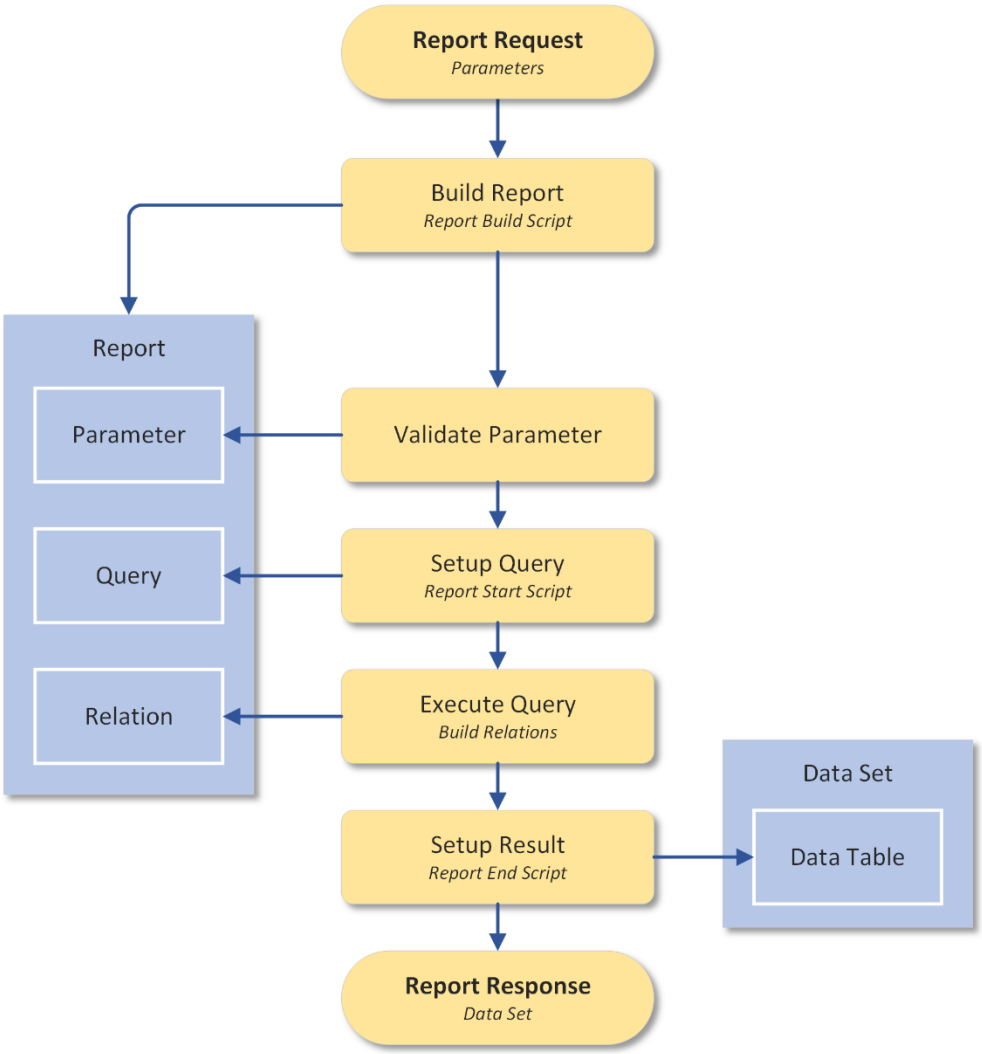


Das sprachabhängige Reportlayout (*ReportTemplate*) ist für die Berechnung der Reportdaten irrelevant und wird vom REST-Client bestimmt (*.frx, .docx, .rdlc, …)*.

Im Reportlayout enthält auch Datenfelder für Schemas, die der REST-Client zur Validierung der generierten Reportdaten (JSON/XML/usw.) verwenden kann.

2.12.1 Reportverarbeitung

Die Erstellung eines Reports erfolgt in folgenden Schritten:



1. Erstellung des Reports: Die Reportparameter können im *Report BuildScript* angepasst werden.
2. Überprüfung und Vervollständigung der Report-Parameter.
3. Vorbereitung der Abfragen (*Queries*): Die Abfragen sind im Report vordefiniert und können im *Report Start Script* angepasst werden.
4. Abfrage der Reportdaten: Der Report kann mehrere Abfragen ausführen, wobei die Daten eines API-Endpunkts (GET) in einer Tabelle (Memory) gespeichert werden. Abschliessend werden die Beziehungen zwischen den Tabellen hergestellt.
5. Weiterverarbeitung der Reportdaten: Im *Report End Script* können die Tabellen an das gewünschte Zielformat angepasst werden. Zur Behandlung komplexer Fälle sind Laufzeitabfragen im Script möglich.

Jeder Schritt ist optional und kann je nach Art des Reports variieren. Ein einfacher Report kann nur aus Abfragen bestehen. Ein komplexer Report kann die gesamte Logik der Nachverarbeitungslogik enthalten.

Weitere Report Funktionen:

- Zugriff auf lokalisierte Texte
- Ausgabe von benutzerdefinierten Attributen
- Aktualisierung der Metadaten des Dokuments, einschliesslich der Abfrageparameter
- Umfangreiche Bearbeitungsmöglichkeiten der Reporttabellen per Scripting (Microsoft https://docs.microsoft.com/en-us/dotnet/api/system.data.dataset)
- Kategorisierung von Reports zur Einschränkung der Auswahl im REST-Client mit Cluster

2.12.2 Report-Logs

Die Ausführung eines Reports kann im Log protokolliert werden und dient als Historie für Reports, die Informationen aus früheren Reports benötigen, wie z.B. das Meldedatum oder den Meldestatus.

3 Basiskonzepte

3.1 Sprachen

Der Payroll Dienst unterstützt alle RFC 4646 Sprachen mit Englisch als Übersetzungsbasis. Für Referenzen zwischen Objekten wird der englische Bezeichner verwendet.

3.2 Audit Trail

Für die zentralen Payroll Daten protokolliert die Engine jede Änderung im Audit-Trail. Dies gilt für alle Regelwerksobjekte und Falldaten. Der Audit-Trail der Falldaten dient auch als Grundlage für die Berechnung der Zeitraumwerte.

3.3 Datentypen

Die Engine bietet folgende Datentypen an:

Datentyp	JSON	C#	Domäne
String	String	String	
WebResource	String	String	RFC 1738
Date, DateTime	String	DateTime	ISO 8601
Integer	Integer	Int32	
Decimal, Money, Percent, Hour, Day, Week, Month, Year, Distance, NumericBoolean	Number	Decimal	
Boolean	Boolean	Boolean	
None	Null	null	

3.4 Kalender

Der Kalender unterteilt die Daten in zwei Zeiteinheiten:

- **Periode** in welcher die Löhne gezahlt wird, in der Regel der Kalendermonat
- **Zyklus** in dem sich die Lohnperioden wiederholen, , in der Regel das Kalenderjahr

Der Lohnzyklus ist immer ein Vielfaches der Lohnperiode. Die folgende Übersicht zeigt die möglichen Kombinationen der beiden Zeiteinheiten mit der Anzahl der Lohnzahlungen innerhalb eines Zyklus:

Periode ↓ / Zyklus →	Woche	Zwei Wochen	Halb-monat	Mond Monat	Monat	Zwei Monate	Quartal	Halbjahr	Jahr
Woche	1	2	-	4	-	-	13	26	52
Zwei Wochen	-	1	-	-	-	-	-	13	26
Halbmonat	-	-	1	-	2	4	6	12	24
Mond Monat	-	-	-	1	-	-	-	-	13
Monat	-	-	-	-	1	2	3	6	**12**
Zwei Monate	-	-	-	-	-	1	-	3	6
Quartal	-	-	-	-	-	-	1	2	4
Halbjahr	-	-	-	-	-	-	-	1	2
Jahr	-	-	-	-	-	-	-	-	1

3.4.1 Kalenderkonfiguration

Die Zeitberechnungen der Engine basieren auf der folgenden Kalenderkonfiguration:

- Zeiteinheit des Zyklus
- Zeiteinheit der Periode
- Zeitwert zyklisch oder periodisch, Beispiele:
  - Zyklus=Jahr, Periode=Monat, Basiswert=60'000 → Lohnwert=5’000
  - Zyklus=Halbjahr, Periode=Zwei Wochen, Basiswert=52'000 → Lohnwert=4’000
- Erster Monat im Jahr für Geschäftsjahre (Default: Januar)
- Regel für die erste Woche im Jahr
- Erster Tag in der Woche (Montag)
- Durchschnittliche Anzahl der Monatstage mit einem Durchschnittswert
- Definition der Tage in der Arbeitswoche (z.B. Montag bis Freitag)

Der Kalender wird durch folgende Objekte bestimmt und kann auf verschiedenen Ebenen übersteuert werden. In der Auswertungspriorität gibt es folgende Kalenderbenutzer:

1. Lohnart (temporär im Lohnlauf)
2. Mitarbeiter
3. Unternehmensbereich
4. Mandant

3.4.2 Weltzeit

Die Datumswerte der REST-API müssen in UTC angegeben werden, andere Zeitmodelle führen zu Fehlern.

3.4.3 Zeittypen

Die Berechnung der Zeitwerte basiert auf folgenden Zeittypen:

Typ	Beschreibung	Perioden beginn	Perioden ende	Anzahl Periodenwerte
Timeless	Wert ohne zeitliche Bindung z.B. Vorsorgepflicht	-	-	1
Moment	Wert, der einem Zeitpunkt zugeordnet wird z.B. Bonus	x	-	1..n Summe

Period	Wert für einen Zeitraum z.B. Arbeitsstunden	x	(x)	n überlagernd
CalendarPeriod	Wert tageweise auf Kalenderperiode verteilt z.B. Monatslohn	x	(x)	n überlagernd

3.5 Objekteigenschaften

Jedes API-Objekt hat die folgenden Grundeigenschaften:

- Identifikation, laufende Nummer innerhalb seines Typs
- Engine-Zeit der Erstellung in UTC (*Created*)
- Engine-Zeit der letzten Änderung in UTC (*Updated*)
- Objektstatus, aktiv (Voreinstellung) oder inaktiv

3.6 Objektattribute

Objektattribute sind benutzerdefinierte Daten, deren Inhalt keinen Einfluss auf das Verhalten der Engine hat. Attribute können für die folgenden Szenarien verwendet werden:

- Referenzen auf externe Objekte in Umsystemen
- Speicherung von Zusatzdaten (wenn kein Zeitwert oder Fallwert)
- Steuerung in Client-Anwendungen, z.B. Steuerung der Eingabe von Falldaten
- Steuerung/Parametrisierung von Scripting-Funktionen (Kollektor oder Lohnart)
- Speicherung zusätzlicher Lohnartenresultate mit z.B. Meldedaten

Die Attribute werden in einer Werteliste (Dictionary) im Objekt verwaltet. Einige Objekte bieten API-Endpunkte an, um die Objektattribute direkt zu bearbeiten.

In REST-Abfragen können Attributfelder zur Filterung verwendet werden (siehe Attributabfragen).
Übersicht der API-Objekte mit Attributen:

Attributobjekt	Attribut API	Audit
Tenant	x	
User	x	
Calendar	x	
Task	x	
Division	x	
Employee	x	
Payroll	x	
Payroll Layer	x	
Payrun Job	x	
Webhook	x	
Regulation	x	
Case		x
Case Field		x
Case Relation		x
Collector		x
Wage Type		x
Report	x	x
Case Value		
PayrollResult		

4 Payroll Tests

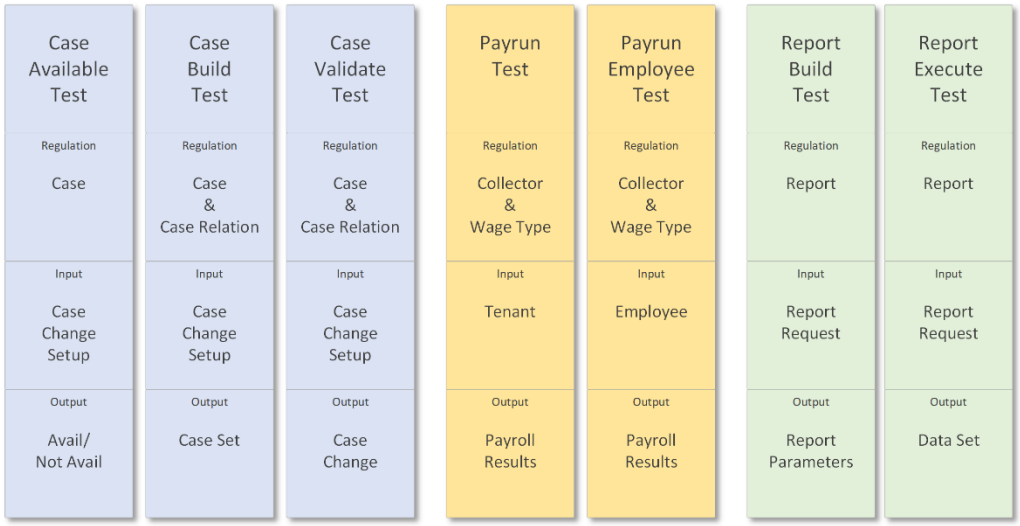
Basierend auf modernen Ansätzen in der Softwareentwicklung, bietet die Payroll Engine auch eine testgetriebene Entwicklung der Lohnberechnung. Mit Hilfe der Client Services (siehe Client Services) können folgende Aspekte im Lohnwesen automatisiert getestet werden:

- Mitarbeiterfälle in der Vergangenheit und der Zukunft
- Beliebige Lohnläufe innerhalb einer Lohnperiode und über mehrere Lohnperioden hinweg
- Vergleich der berechneten Lohndaten mit den erwarteten Ergebnissen

Automatisierte Tests sind in drei Bereichen möglich:

- Testen eines Falles
- Testen eines Lohnlaufes für einen Mandant oder einen Mitarbeiter
- Testen eines Reports

Alle Objekte des Regelwerks können mit den folgenden Tests automatisiert getestet werden:



4.1 Fall Tests

Fall-Tests können verwendet werden, um die Verfügbarkeit, Generierung und Validierung eines Geschäftsfalls zu testen. Die Testfälle können durch Konfiguration (JSON) oder durch Programmierung (C# mit Client Services) definiert werden.

4.2 Lohnlauf Tests

4.2.1 Mandantenlohnlauf-Tests

Beim Mandanten-Lohnlauftest wird ein neuer Mandant angelegt und die Ergebnisse der Lohnläufe mit den erwarteten Ergebnissen verglichen. Nach dem Test wird der Mandant gelöscht.

4.2.2 Mitarbeiterlohnlauf-Tests

Für umfangreichere Mandantenszenarien sind Tests für einzelne Mitarbeiter möglich. Der Test des Mitarbeiterlohnlaufs setzt einen bestehenden Mandanten voraus. Der Lohnlauf wird auf der Kopie eines bestehenden Mitarbeiters durchgeführt. Die Bereinigung der Testdaten erfolgt manuell.

4.3 Report Tests

Für die Payroll Reports kann sowohl die Reporterstellung (Report Parameter) als auch die Reportausführung getestet werden. Die Reporttests können durch Konfiguration (JSON) oder durch Programmierung (C# mit Client Services) definiert werden.

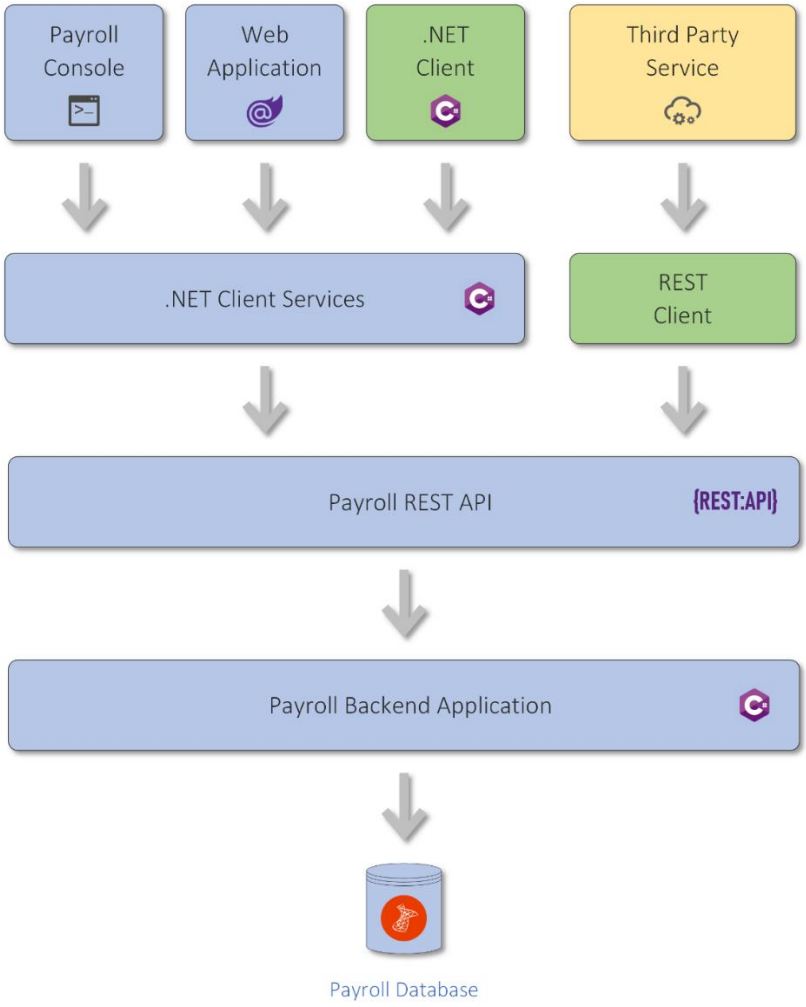
4.4 Tests anwenden

Der Test wird in JSON definiert und mit der Anwendung Payroll Console ausgeführt. Die Testergebnisse werden auf dem Bildschirm angezeigt und Testfehler in Logdateien protokolliert.

Die Testfunktionen stehen auch als Programmierbausteine zur Verfügung (siehe Client Services) und können individuell erweitert werden.

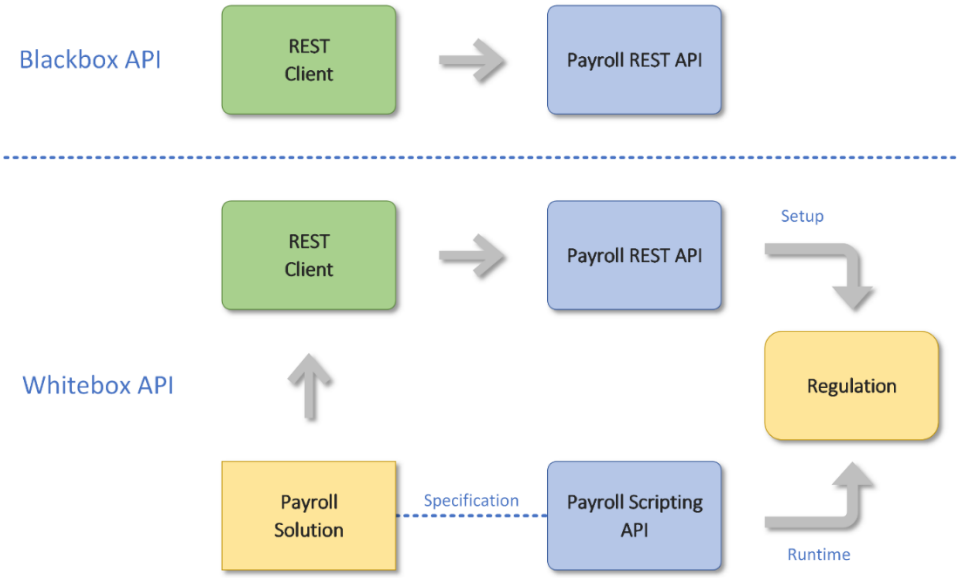
5 Payroll Backend

Das Payroll Backend besteht aus den folgenden Komponenten:



5.1 Whitebox API

Die Payroll Engine ist eine Whitebox REST API mit einer zusätzlichen Schnittstelle, der Scripting API, über die das Laufzeitverhalten beeinflusst werden kann. Die folgende Abbildung zeigt die Unterschiede zwischen der Blackbox und der Whitebox REST API:



5.2 Technologien

Die folgenden Technologien werden im Payroll Backend verwendet:

- REST API OpenAPI 3.0 (OAS3)
  Media type application/json
- Scripting API C# 11
- Server-Applikation .NET 7 Runtime für Server Apps
- Datenbank SQL-Server 2017

5.3 Listenabfragen

Die folgenden Kriterien können zur Einschränkung der Daten bei API-Abfragen verwendet werden:

- Status nach aktiven oder inaktiven Objekten
- Filterausdruck nach OData
- Reihenfolge der Felder nach OData
- Feldauswahl nach OData
- Seiteneinschränkung mit der Anzahl der Objekte pro Seite und dem Seitenoffset

5.3.1 Attributabfragen

Die Engine bietet die Möglichkeit, Listenabfragen nach Attributwerten zu filtern, zu sortieren und zu gruppieren. Dabei bestimmt das Präfix im Attributnamen den Datentyp des Attributs:

- TextAttribut: *TA_*
  →Beispiel:*TA_MyName*
- Datum Attribut: *DA_*
  → Beispiel:*DA_ProjectCreated*
- ZahlAttribut: *NA_*
  →Beispiel:*NA_SecurityLimit*

5.4 Webhooks

Der Engine bietet Webhooks für folgende Aktionen an:

Aktion	Auslöser	Webhook Message	Track
CaseFunctionRequest	*Case* Funktion Anfrage	*Operation: <Custom>* *Request: <Custom>*	-
CaseValueAdded	Neue Case Mutation	*Request: CaseChange*	✓
PayrunFunctionRequest	*Payrun*-Funktion Anfrage	*Request: <Custom>* *Operation: Funktionsname*	-
PayrunJobStarted	Gesetzlicher Lohnlauf-Job freigegeben zur Verarbeitung	*Request: PayrunJob*	✓
PayrunJobCompleted	Gesetzlicher Lohnlauf-Job wurde beendet	*Request: PayrunJob*	✓
ReportFunctionRequest	*Report* Funktion Anfrage	*Operation: <Custom>* *Request: <Custom>*	-
TaskChange	Task Änderung	*Request: TaskChange*	

Mittels HTTP POST wird die Webhook Methode aufgerufen und erhält als Argument eine *Webhook Message*. Aus der Webhook-Antwort wird im Backend der HTTP-Statuscode sowie der Inhalt (JSON) im Backend ausgewertet. Spezifische Webhook Messages werden im Backend getrackt und können über die REST API abgefragt werden.

5.5 Administration API

Die Payroll API bietet verschiedene Endpunkte für die Systemadministration:

- Beenden der Anwendung
- Bereinigen des Anwendungscaches
- Ermitteln der verfügbaren Report-Endpunkte ermitteln (siehe Report)

5.6 Applikation-Logs

Zur Systemanalyse protokolliert die Engine bestimmte Ereignisse in Logdateien, die keine sensiblen Informationen beinhalten. Die Log-Dateien werden im Windows-Systemverzeichnis der Programmdaten (*%ProgramData%*) gespeichert.

Neben dem Applikations-Log gibt es auch kundenspezifische Logs:

- Log für Lohnläufe (siehe Lohnlauf-Logs)
- Log für Reports (siehe Report-Logs)

6 Scripting API

Die Scripting API ermöglicht es dem Benutzer das Laufzeitverhalten der Engine zu beeinflussen:

Fall- Management

- Verfügbarkeit eines Falls bestimmen
- Aufbau eines Falls und Fallwerte bestimmen
- Verhalten zwischen Fällen festlegen
- Validierung von Fällen

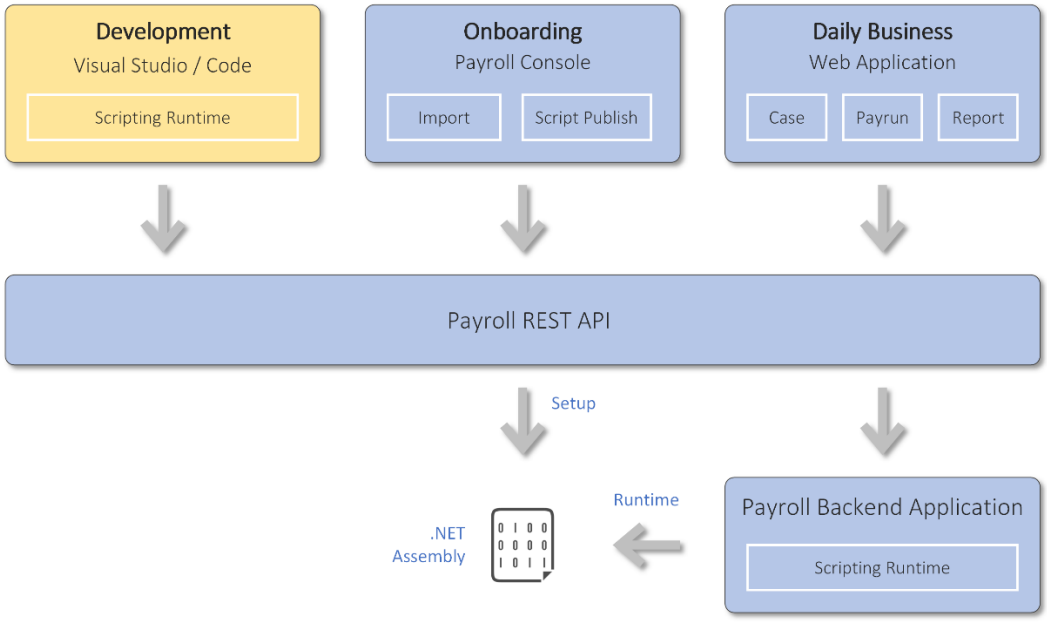
Lohnlauf

- Mitarbeiter und Lohnarten einschränken
- Lohn mit Kollektoren und Lohnarten berechnen und speichern
- Kommunikation mit externen Diensten (Webhooks)
- Zusätzliche Lohndaten berechnen und speichern

Report

- Reportparameter festlegen
- Reportdaten berechnen

Die folgende Abbildung zeigt den Zusammenhang zwischen der Payroll REST-API und der Scripting-API:

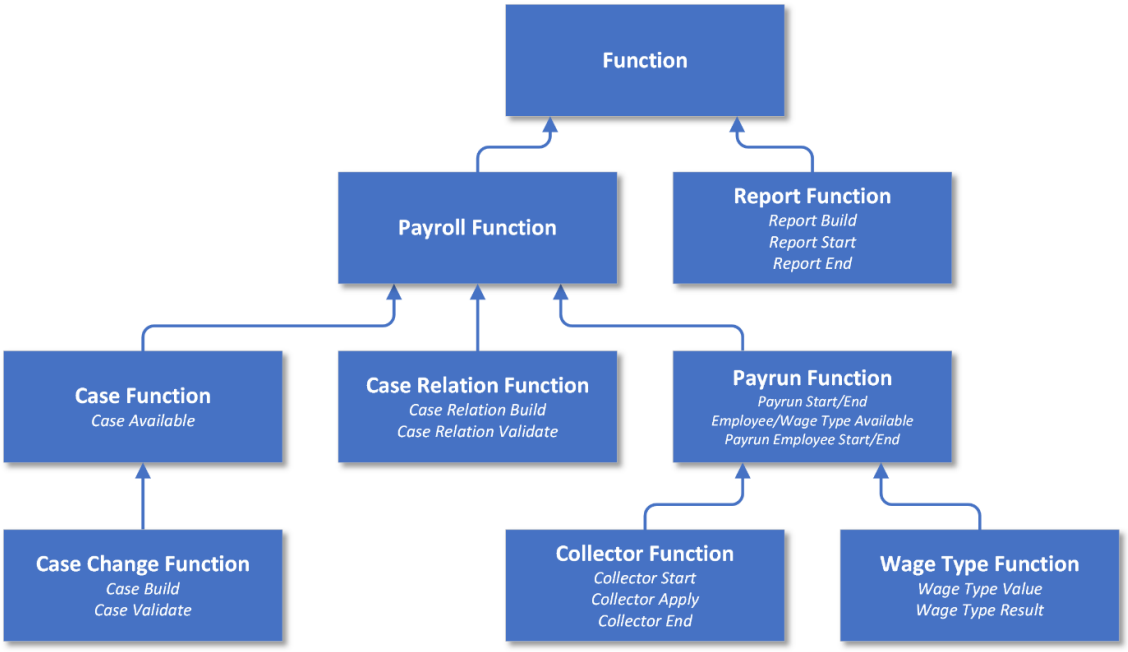


Beim Onboarding werden Scripts entwickelt und mit der *Payroll Console* zur Payroll REST API übertragen. Beim Einspielen der Scripts werden diese in die Maschinensprache übersetzt (kompiliert). Zur Laufzeit werden die binären Scripts vom Payroll Prozess geladen und ausgeführt. Dabei kann der Script-Code die Funktionen der Scripting-API nutzen.

In der Scripting Entwicklung können zur Fehlersuche Scripts lokal getestet werden (Debugging).

6.1 Funktionen

Eine Funktion ist eine programmierte Sequenz, die von einem Engine-Prozess an bestimmten Stellen aufgerufen wird. Die Funktionen sind in folgende Bereiche unterteilt:



Die verfügbaren Scripting Funktionen sind:

Objekt	Funktion	Beschreibung	Typ	Überlagerung
Case	Available	Testen, ob ein Fall verfügbar ist	bool?	1. Top-Down
Case	Build	Fall erstellen	bool?	1. Top-Down
Case	Validate	Testen, ob ein Fall gültig ist	bool?	1. Top-Down
Case Relation	Build	Fallbeziehung erstellen	bool?	1. Top-Down
Case Relation	Validate	Testen, ob eine Fallbeziehung gültig ist	bool?	1. Top-Down
Payrun	Start	Lohnlauf Beginn	bool?	-
Payrun	EmployeeAvailable	Testen, ob der Mitarbeiter verfügbar ist	bool?	-
Payrun	EmployeeStart	Lohnlauf Mitarbeiter Start	bool?	-
Payrun	WageTypeAvailable	Testen, ob eine Lohnart verfügbar ist	bool?	-
Payrun	Employee End	Lohnlauf Mitarbeiter Ende	-	-
Payrun	End	Lohnlauf Ende	-	-
Collector	Start	Kollektor Beginn	-	1. Top-Down
Collector	Apply	Lohnart Wert anwenden	decimal?	1. Top-Down
Collector	End	Kollektor Ende	-	1. Top-Down
Wage Type	Value	Wert der Lohnart berechnen	decimal?	1. Top-Down
Wage Type	Result	Ergebnisse der Lohnart bestimmen	-	* Bottom-Up
Report	Build	Report aufbereiten	bool?	-
Report	Start	Beginn der Reporterstellung	-	-
Report	End	Ende der Reporterstellung	-	-

Die Überlagerung “1. Top-Down“ stellt sicher, dass die erste Funktion, die einen Wert liefert, von den Abrechnungsregelwerken verwendet wird. Wenn ein undefinierter Wert zurückgegeben wird, wird die Funktion des zugrundeliegenden Regelwerks ausgeführt.

Bei der “* Bottom-Up“ Überlagerung werden die Daten, beginnend mit der untersten Regelwerksschicht, zusammengeführt und es besteht die Möglichkeit, Werte im übergeordneten Regelwerk zu überschreiben.

Im Kapitel Payroll Script sind beide Szenarien dargestellt.

6.1.1 Funktionen erweitern

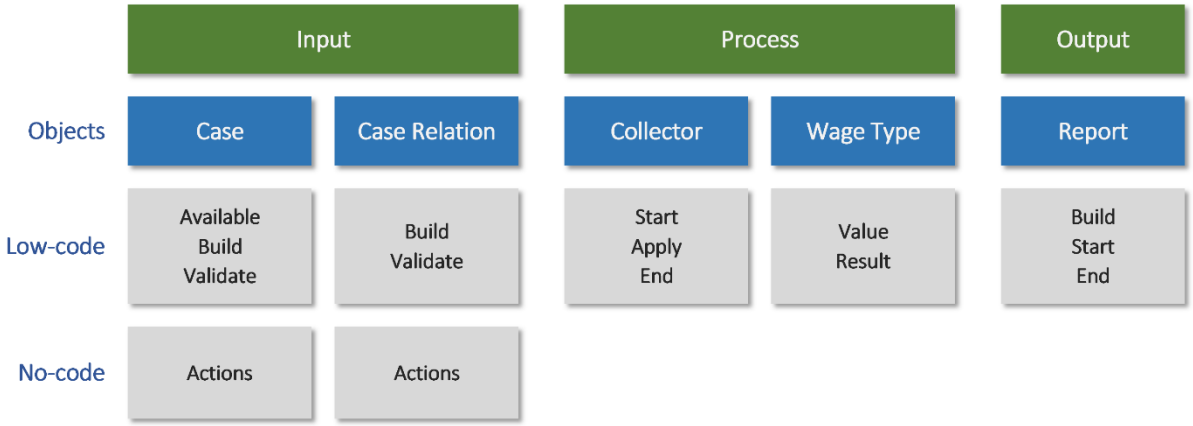
Die integrierten Funktionen können durch Skripte erweitert werden. Dazu gibt es zwei Möglichkeiten in C#:

- Erweiterungsmethode (Microsoft Extension Methods)
- Partielle Klassen (Microsoft Partial Classes)

Die partiellen Klassen eignen sich zur Auslagerung des Business Codes in ein eigenes Klassenmodell. Im Case Scripting bieten Extension-Methoden die Möglichkeit, die Erzeugung und Validierung pro Case Field zu definieren.

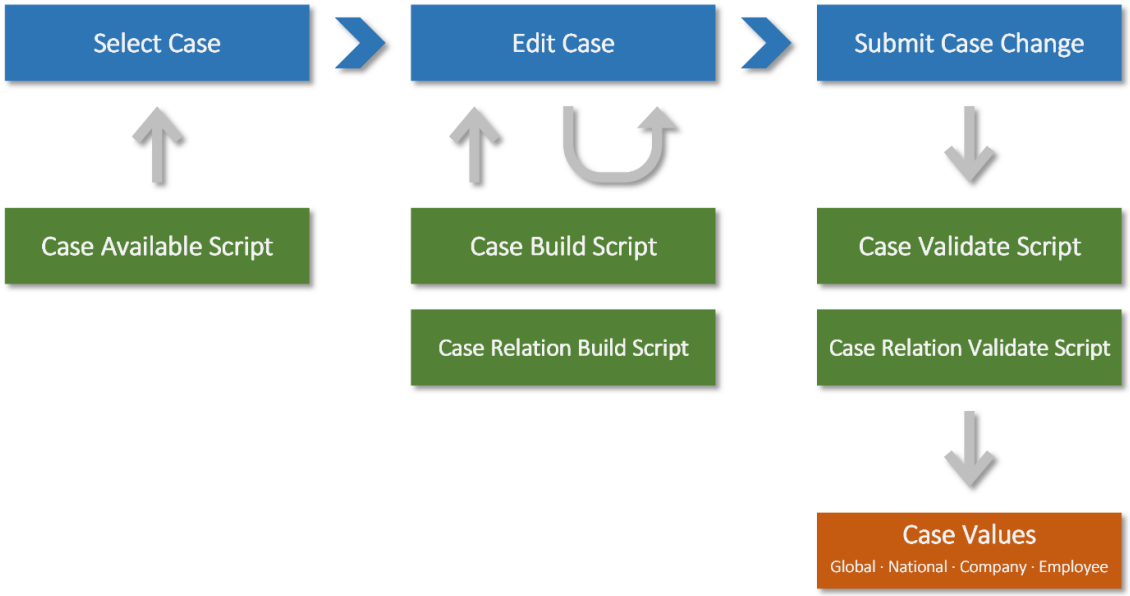
6.2 Regelwerk Scripting

Zur Steuerung des Laufzeitverhaltens bietet die Payroll für die Regelwerksobjekte Scripting-Funktionen an, welche mittels C# Low-Code beschrieben werden. Für das Case Management können die Funktionen über No-Code (*Actions*) gesteuert werden:



6.3 Case Scripting

Der Lebenszyklus eines Lohnfalls besteht aus den Phasen Verfügbarkeit, Generierung und Übermittlung. In der Generierungsphase wird der Fall erstmalig erstellt und bei jeder Datenänderung neu aufgebaut. Daraus ergibt sich folgende Bearbeitungsreihenfolge:

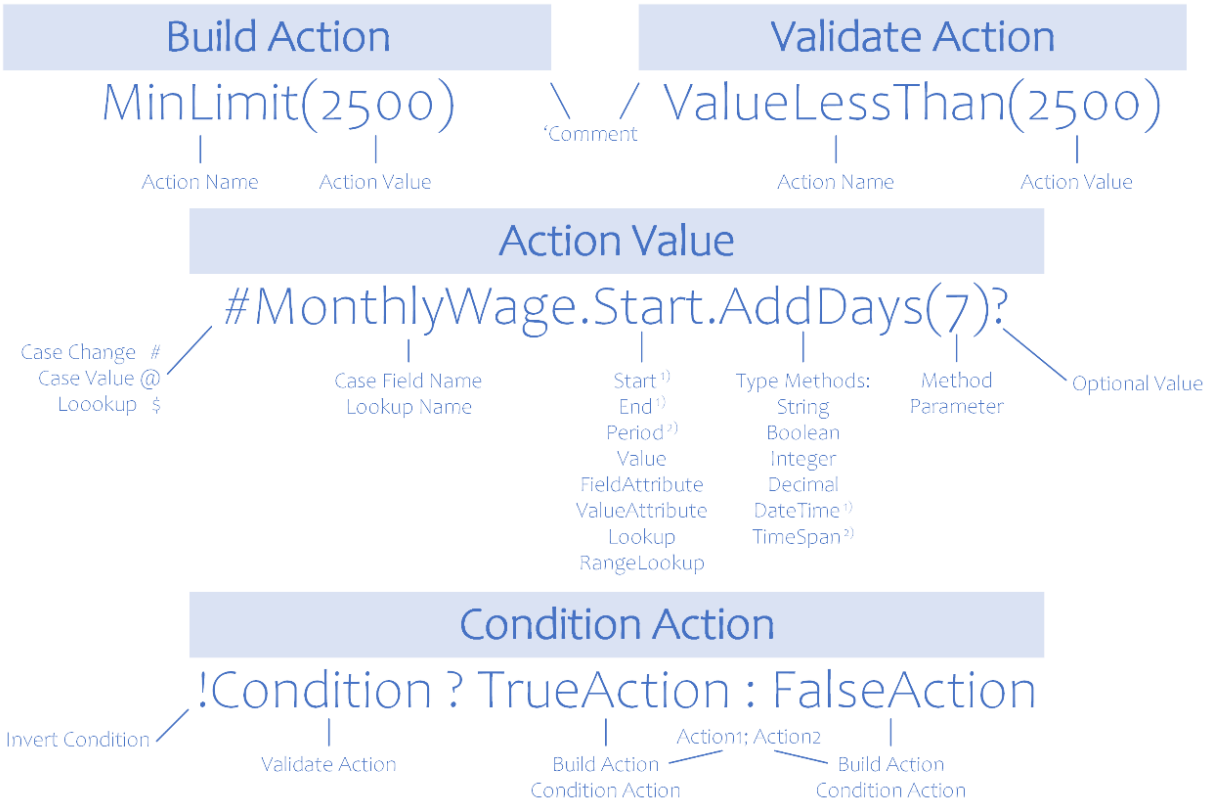


6.3.1 Case Actions

Case Actions bieten die Möglichkeit, die Fallfunktionen ohne Programmierkenntnisse zu steuern. Case Actions bieten die folgenden Funktionen:

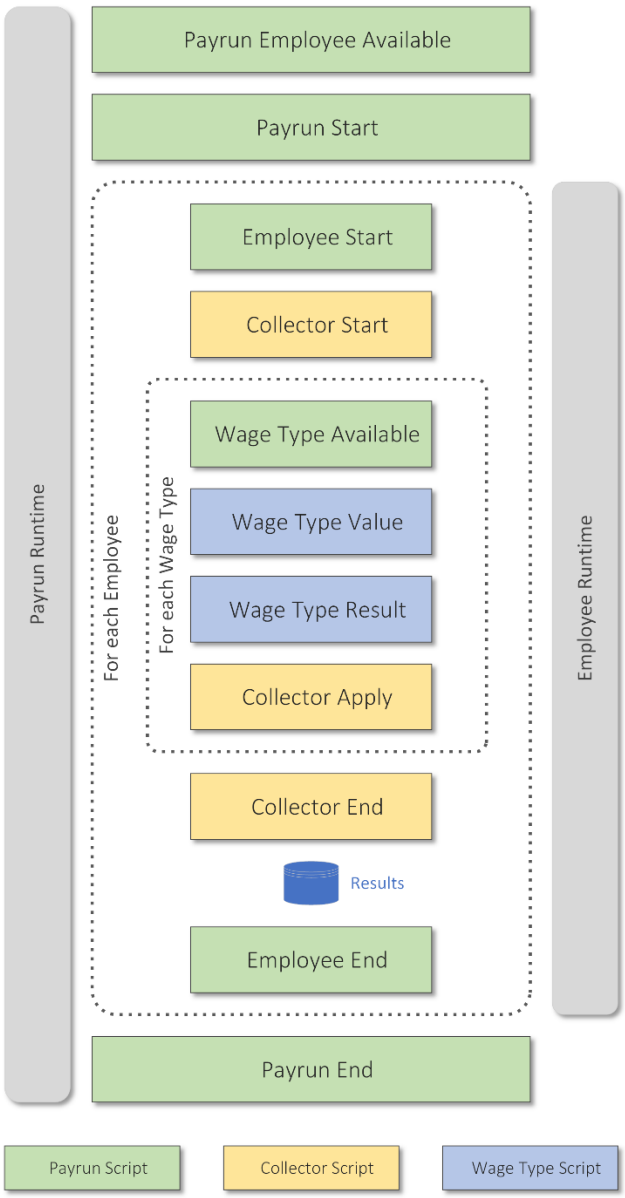
- Zugriff auf Mitarbeiterdaten und Lookups
- Werte bedingt setzen
- Werte prüfen
- Steuerung von Eingabefeldern

Die folgende Übersicht zeigt den Befehlssatz von Case Actions:



6.4 Lohnlauf Scripting

Den Lohnlauf steuern folgende Scripting Funktionen:



Während des Lohnlaufes können Werte zwischen den Funktionen ausgetauscht werden. Dabei wird zwischen Lohnlauf-Laufzeitwerten (*PayrunRuntime*) und Mitarbeiter-Laufzeitwerten (*EmployeeRuntime*) unterschieden. Die Lohnlauf-Laufzeitwerte bestehen für den gesamten Lohnlauf, die Mitarbeiter-Laufzeitwerte bestehen jeweils pro Mitarbeiter. Am Ende des Lohnlaufes stehen alle Laufzeitwerte für die weitere Verarbeitungen zur Verfügung.

Übersicht der Lohnlauf-Laufzeitwerte:

Funktion	Lohnlauf-Laufzeitwerte	Mitarbeiter-Laufzeitwerte
Payrun	Zugriff	-
Collector	Zugriff (vererbt)	Aktueller Mitarbeiter
Wage Type	Zugriff (vererbt)	Aktueller Mitarbeiter
Payrun End	Zugriff (vererbt)	Alle Mitarbeiter

Ist die Lohnart einem oder mehreren Kollektoren zugeordnet, wird dessen Wert (*Wage Type Value/Result*) auf die Kollektoren angewendet (*CollectorApply*).

6.5 Lohnlauf Zeitwerteberechnung

Bei der Berechnung eines Zeitraumwertes werden alle relevanten Mutationen innerhalb des Periodenzeitraumes anteilsmässig gemäss Kalender aufgeteilt (z.B. untermonatige Lohnanpassungen). Die Scripting API bietet die Möglichkeit, die Verteilungslogik des Kalenders zu nutzen, was die Berechnungssyntax erheblich vereinfacht. Zwei Fallwerte mit unterschiedlichen Mutationen können mit den üblichen mathematischen Operatoren (Plus, Minus, Multiplikation und Subtraktion) berechnet werden.

6.6 Lohnlauf-Logs

Über Scripting-Funktionen können Log-Einträge auf verschiedenen Ebenen erzeugt werden. Die Logs werden dem Mandanten zugeordnet und können über die API ausgewertet werden. Über einen Startparameter des Lohnlauf-Jobs kann festgelegt werden, welche Log-Ebenen protokolliert werden sollen.

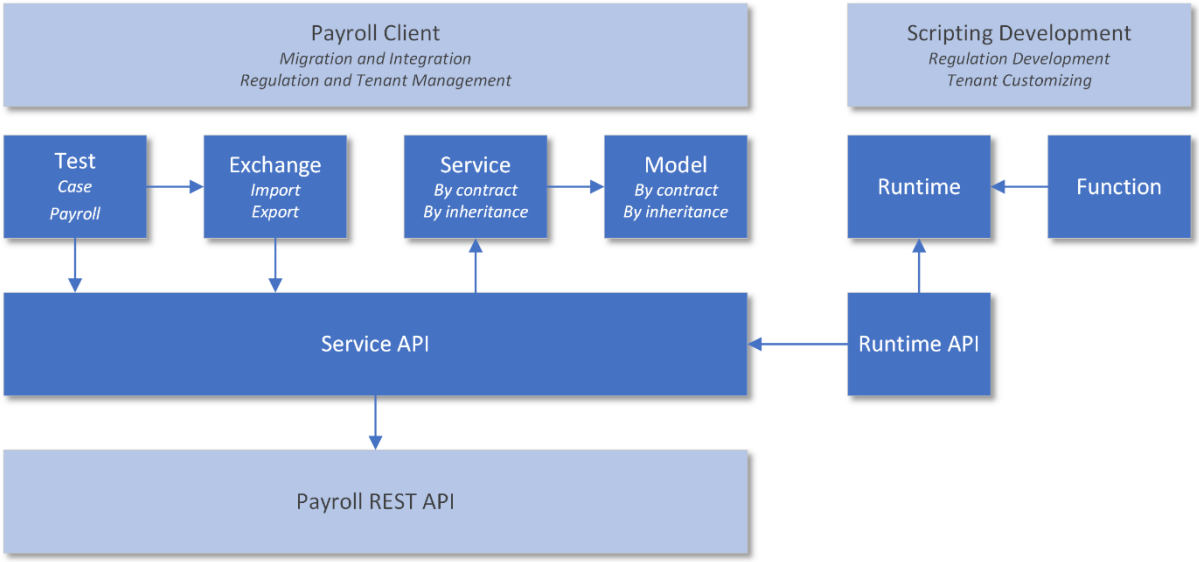
6.7 Scripts in der REST API

Scripts werden als Objektwerte geführt (z.B. Lohnart *ValueExpression* und *ResultExpression*) und werden beim Einlesen des Objekts (POST) in Maschinencode kompiliert. Im Falle eines Syntaxfehler gibt der Payroll Dienst den Fehlercode “422 Unprocessable Entity“ zurück.

7 Client Services

Die Client Services sind zusätzliche Dienste, um die Payroll API effizienter zu nutzen für

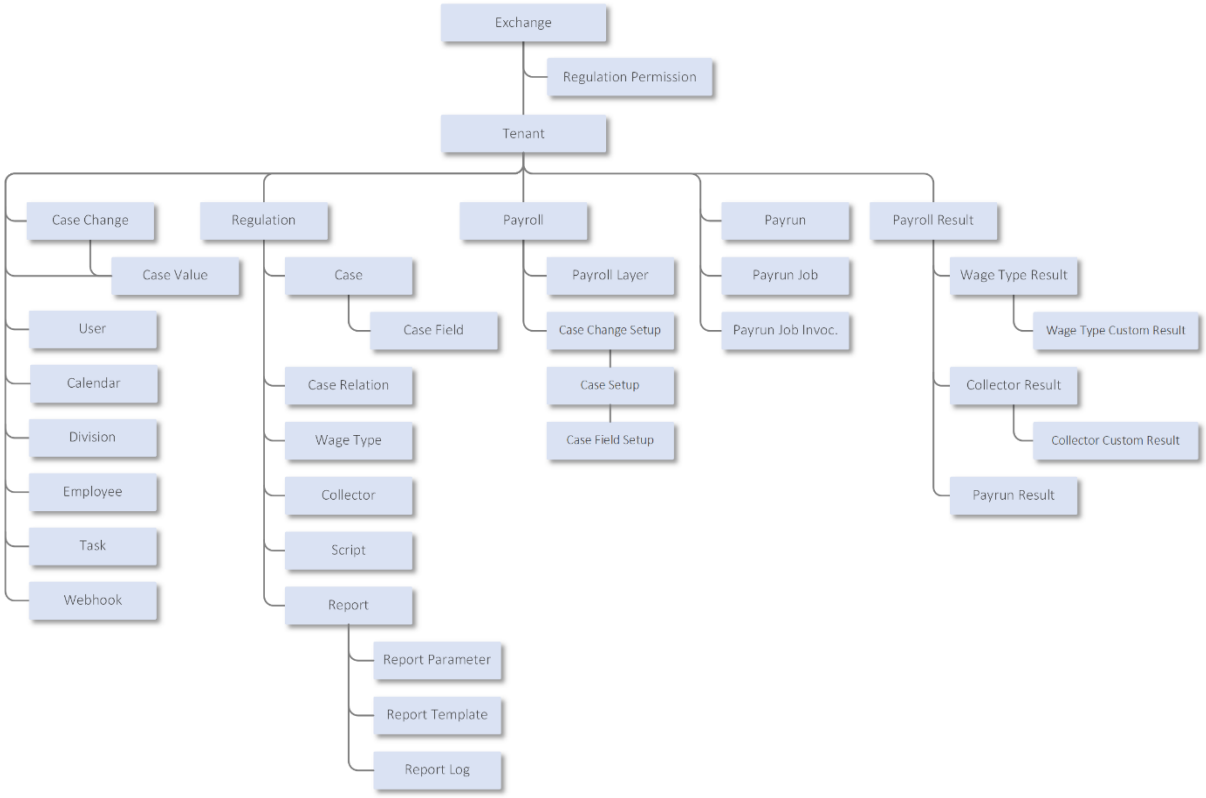
- Migrationen und Integrationen (z.B. Schnittstellen)
- zur Verwaltung von Regelwerken und Mandanten
- das Testen von Regelwerken
- die Entwicklung von Scripts



Model	Model der API-Objekte (Swagger-Schema) erweitert um Exchange-relevante Werten und Eigenschaften. Ein Model-Objekt ist kopier-/vergleichbar und kann durch Vererbung oder Contract/Interface verwendet oder erweitert werden.
Service	Zugriff auf die API-Endpunkte (Swagger-Endpunkte). Ein Service kann durch Vererbung oder Contract/Interface genutzt oder erweitert werden.
Exchange	JSON Import und Export von Payrolls.
Test	Tests von Geschäftsfällen und Lohnläufen.
Service API	Kommunikation zur Payroll API.
Function	Vorlagen zur Entwicklung von Funktions-Scripts.
Runtime	Funktion Laufzeitumgebung zur Entwicklung von Funktions-Scripts.
Runtime API	Kommunikation zur Service API.

7.1.1 Exchange Modell

Das Exchange Modell enthält alle API-Objekte der Payroll Engine für den Datenaustausch:

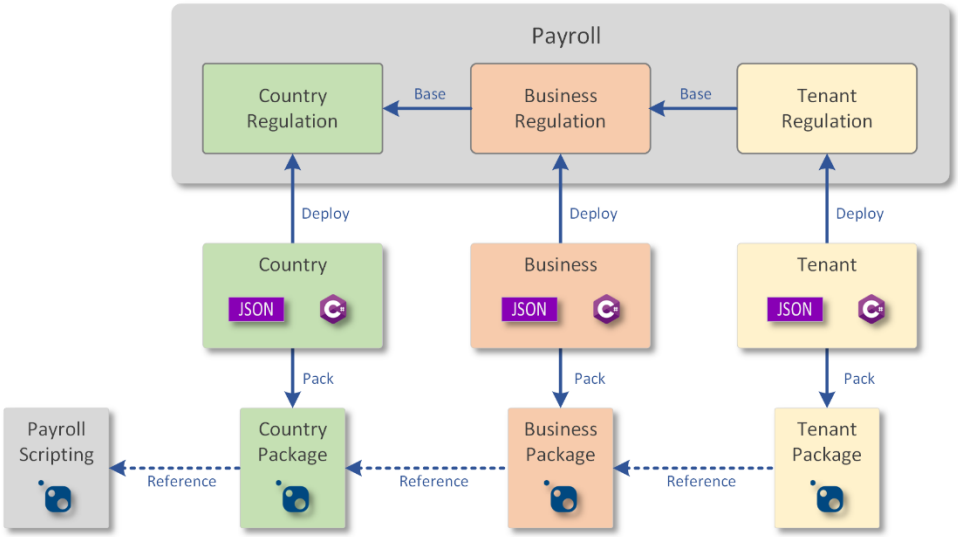


Für die Bearbeitung des Modells stehen verschiedene Werkzeuge zur Verfügung (z.B. Import und Export), die auch für reduzierte Modellbereiche funktionieren. Darüber hinaus gibt es Klassen, die die Abfolge der Modellverarbeitung übernehmen und dem Anwender den direkten Objektzugriff ermöglichen.

7.1.2 Regelwerke entwickeln und bereitstellen

Regelwerke sind in JSON Dateien beschrieben und beinhaltet die programmierte Lohnlogik als C# Code. Dieser wird als eingebettete Objekt-Expression oder in einer Quellcodedatei (.cs) bestimmt. Die Auslagerung der Programmlogik in Quellcodedateien wird für geteilte Regelwerke vorausgesetzt. Neben der Dokumentation wird der NuGet-Mechanismus verwendet, um versionierte Regelwerke bereitzustellen.

Die folgende Abbildung zeigt die Verwendung und Bereitstellung von Regelwerk-Packages, sowie die Bereitstellung von Regelwerken für das Lohnlaufsystem:



8 Clients

Basierend auf den Client Services bestehen zum Backend folgende Clients:

- Payroll Konsole: Importieren, Exportieren und Testen von Payrolls
- Web Applikation: Visuelle Verwaltung der Payroll Engine

8.1 Payroll Konsole

Die Anwendung Payroll Konsole bietet verschiedene Funktionen der Client Services in einer Konsole an:

- Payroll Import (JSON)
- Payroll Export (JSON)
- Payroll Test (JSON: Import Payroll, Lohnlauf, Resultat-Test, Cleanup)
- Mitarbeiter Test (JSON: Import Falldaten, Lohnlauf, Resultat-Test)
- Scripting Import, Export und Rebuild
- Payroll Log Trail
- Payroll-API Abfragen
- Payroll Report in verschiedenen Formaten

8.2 Web App

Die Web App bietet verschiedene Funktionen der Client Services in einer Web Applikation:

- Verwaltung von Mandanten und Mitarbeitern
- Erfassung von Falldaten
- Starten von Lohnläufen
- Auswertung der Lohnlaufdaten
- Verwaltung von User-Tasks und Entwickler-Logs

9 Weitere Informationen

Payroll Engine Website
https://payrollengine.org/

9.1 Open Source

Die Payroll Engine ist ein Open Source Projekt, welches auf GitHub gehostet wird:
https://github.com/Payroll-Engine