Payroll Engine Whitepaper

1 Einführung

Dieses Dokument beschreibt die Funktionsweise der Payroll Engine mit den zugrundeliegenden Konzepten. Kapitel 1 bis 4 beschreiben die Engine aus Business-Sicht. Die Kapitel 5 bis 8 zeigen die technischen Aspekte der Lösung auf.

Mit der Payroll Engine können Lohndaten von Firmen berechnet werden. Im Gegensatz zu herkömmlichen Payroll Systemen unterscheidet sich die Payroll Engine in folgende Themen:

Fallgetrieben	Die Datenänderungen erfolgen fallgetrieben und sind der Zeitachse zugeordnet. Dadurch sind die Geschäftsdaten zu jedem Zeitpunkt gültig. Automatisierte Lohnläufe inklusive Retro- und Forecast-Szenarien sind jederzeit möglich.
Globalisierung	Im gleichen Mandanten können die Geschäftsdaten eines Konzerns mit deren jeweiligen länder- und firmenspezifischen Gesetzen und lokalisierten Reglementen abgerechnet werden.
Regulierungen	Nebst Landesregulierungen (z.B. Swissdec für die Schweiz) können auch als zusätzliche Regulierungsschicht integriert werden, wie z.B. allgemein verbindliche Gesamtarbeitsverträge oder Pensionskassen-Reglemente.
Kundenskalierung	Kundenanpassungen werden ebenfalls in Regulierungen geführt und ermöglicht skalierbare Lösungen für Klein- bis Grossunternehmen.
Anpassungsfähigkeit	Konfigurierbares und erweiterbares Daten- und Verarbeitungsmodell mittels REST API und Software-Entwicklungskit (SDK).

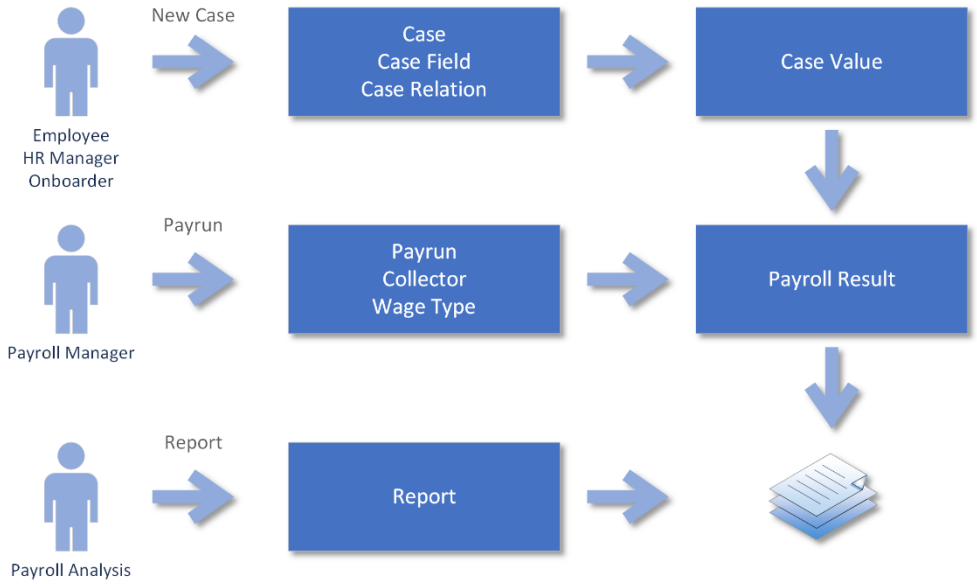
2 Domänenmodell

2.1 Kern Anwendungsfälle

Die zentralen Payroll Anwendungsfälle decken folgende Bereiche ab:

- Case Management: Änderung von Firmen- und Mitarbeiterdaten
- Payrun: Ausführung eines Lohnlaufes
- Report: Auswertung von Lohndaten

Die folgende Abbildung zeigt die Payroll Objekte der zentralen Anwendungsfälle:

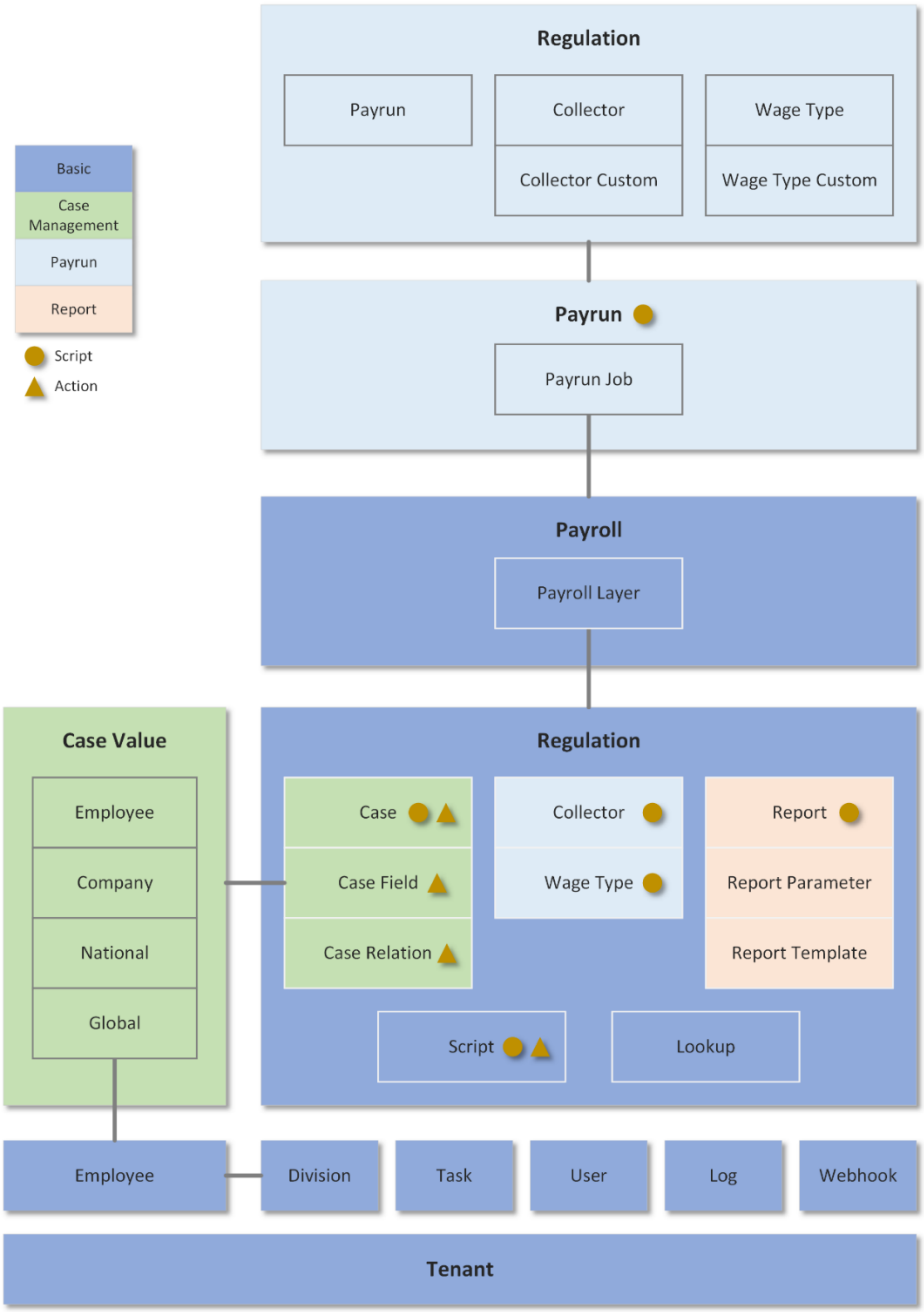


2.2 Modelübersicht

Das Domain Model beinhaltet Objekte für folgende Funktionsbereiche:

- Basis-/Infrastruktur Objekte
- Das Case Management Objekte für die Dateneingabe
- Die Payrun Objekte zur Verarbeiten der Lohndaten
- Reporting Objekte zur Umwandlung der Ausgabedaten

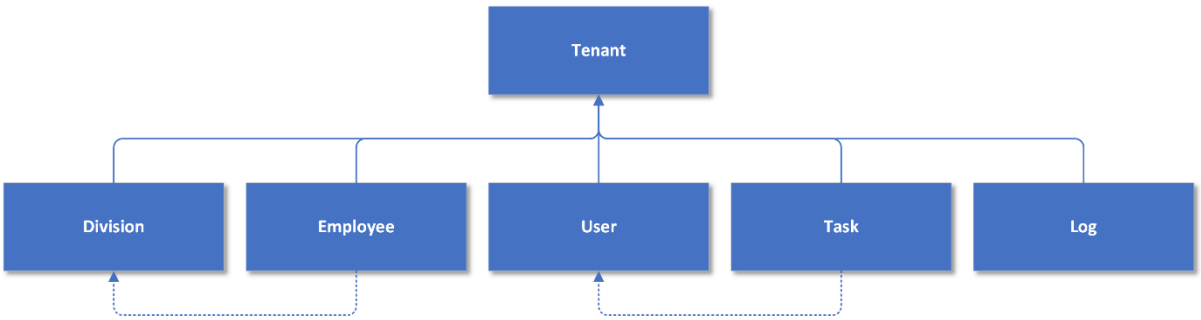
Übersicht Payroll Engine Modell:



Die Zusammenstellung zeigt die statische Topologie auf 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 und Anwender:

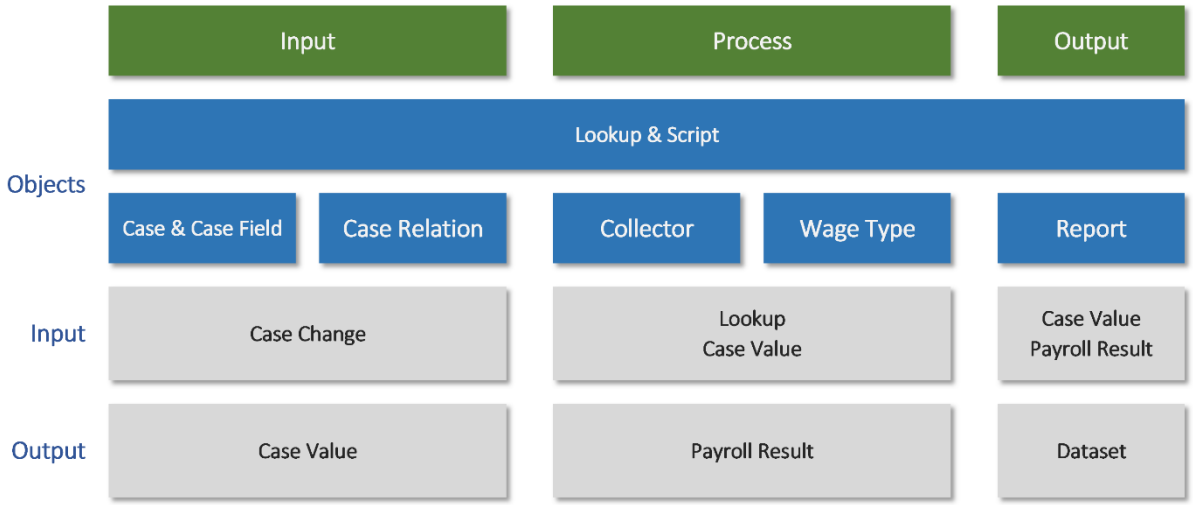


Tenant	Beinhaltet alle Mandatsdaten.
Division	Unterteilt das Unternehmen in Unternehmensbereiche. Jeder Mandant besteht aus mindestens einem Unternehmensbereich.
Employee	Mitarbeiter welche einem oder mehreren Unternehmensbereichen zugeordnet sind.
User	Anwender der Systems.
Task	Die Anwendertasks.
Log	Die Mandantenspezifischen Logeinträge.

Mit Ausnahme der Geteilten Regulation (siehe Kapitel *Geteilte Regulierungen*), sind alle im folgenden beschriebenen Domänenobjekte dem Mandanten zugeordnet.

2.4 Regulierung

Die Regulierung beinhaltet die Definition des Lohnes, mit den Elementen zur Eingabe, Verarbeitung und Ausgabe von Lohndaten:

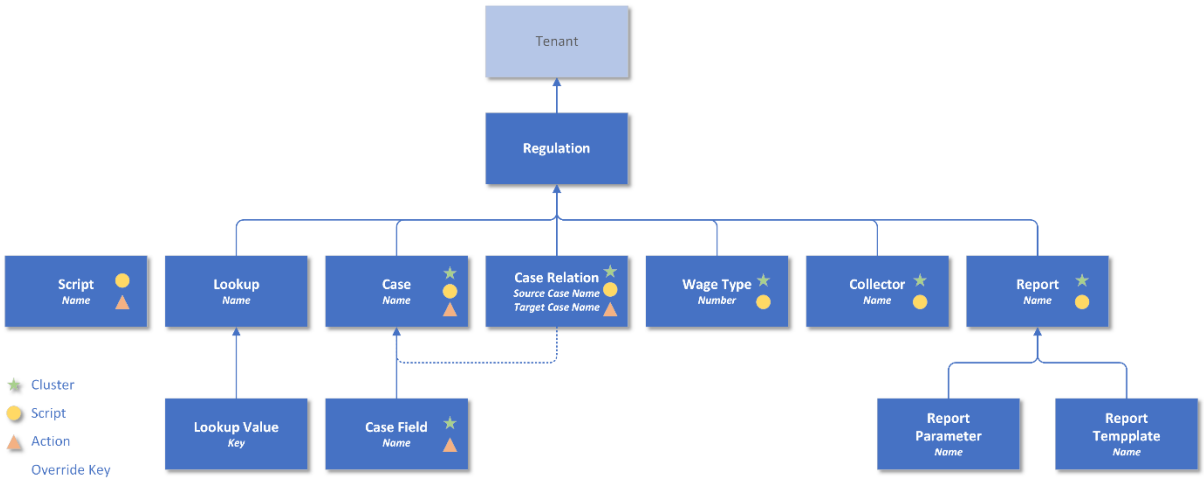


Die Regulierung deckt folgende Themenbereiche ab:

- Bestimmung des Datenmodells und den Geschäftsregeln
  siehe Kapitel *Case und Case Fields*
- Berechnung der Lohndaten
  siehe Kapitel *Lohnlauf*
- Auswertung der Firmen- und Mitarbeiten-Daten
  siehe Kapitel *Report*
- Testen der Firmen- und Mitarbeiten-Daten
  siehe Kapitel Payroll *Testing*
- Bestehende Regulierungen anpassen
  siehe Kapitel *Vererbung von Regulierungen*
- Software zwischen Mandaten teilen
  siehe Kapitel *Geteilte Regulierungen*
- Software Deployment
  siehe Kapitel *Regulierungen entwickeln und bereitstellen*

2.4.1 Regulierungsobjekte

Die Regulation beinhaltet folgende Objekte:



Regulation	Beinhaltet die Daten einer Regulierung.
Script	Geteilter Code von den Funktionen innerhalb der Regulierung.
Lookup **Lookup Value**	Auswahldaten für Anwendungen (z.B. Geschlecht) und Wertetabellen für Scripting-Funktionen (z.B. Quellensteuer).
Case	Lohn Anwendungsfall in den Bereichen Global, National, Unternehmen und Mitarbeiter.
Case Field	Das Feld welchen den Fallwert beschreibt. Mitarbeiter Falldaten können für einen oder alle Unternehmensbereiche gelten (siehe Kapitel *Division*).
Case Relation	Regelt die Abhängigkeit zwischen zwei Fällen.

Wage Type	Bestimmt die Lohnart und deren Berechnung. Die Lohnarten werden in der Reihenfolge der Lohnartnummer einem Dezimalwert (z.B. 1000) berechnet. Durch Nachkommastellen in der Lohnartnummer (z.B. 1000.1) lassen sich untergeordnete Lohnarten in die Bearbeitungsreihenfolg einbinden. Die Lohnart bestimmt welche Kollektoren oder Kollektorgruppen relevant sind.
Collector	Aggregiert die Lohndaten während des Lohnlaufs für z.B. die Lohnbasen. Die Art des Kollektors wird durch den Aggregationstyp z.B. Summierung, Minimum, Maximum, Zählen usw. gesteuert. Kollektoren können einzeln oder in Gruppen ausgeführt werden.
Report	Definition des Reports.
Report Parameter	Die Parameter des Reports.
Report Template	Das sprachbezogene Vorlage des Reports.

Mittels Überlagerungen (Derived) kann ein Regulierungsobjekt von einer übergeordneten Branchen- oder Kundenregulierung übersteuert werden (siehe Kapitel *Payroll*). Die Erkennung welches Objekt zu übersteuern ist, erfolgt mit dem Überlagerungsschlüssel, z.B. der Name des Falls. Clusterobjekte bieten einen erweiterten Tagging-Mechanismus an, um gleichartige Objekte nach Anwendungskriterien zu berücksichtigen oder auszugrenzen (siehe Kapitel Payroll Clusters).

Zu jedem Regulierungsobjekt existiert ein Audit-Trail (siehe Kapitel *Audit Trail*) mit entsprechenden REST Endpunkten. Änderungen einer Regulierung (z.B. Jahresaktualisierung) werden in Versionen geführt, wobei jede Version ab einer Lohnperiode gilt. Dies gewährleistet die korrekten Zeitdaten Lohnläufe mit Rückrechnung oder Forecast.

2.4.2 Anwendungsszenarien

Für die Payroll Regulierung sind folgende Einsatzgebiete möglich:

- Für einen Mandanten spezifische Regulierung im Onboarding
- Dienstleister bietet eigenen kommerzielle Regulierungen für seinen Kundenkreis
- Branchen Regulierungen wie z.B. Verbandslösungen
- Landesspezifische und gesetzliche Regulierungen wie z.B. Swissdec
- Unternehmens-Regulierungen als länderübergreifende Grundlage
- Regulierungen für Tests und Weiterentwicklungen

2.4.3 Vererbung von Regulierungen

Die Regulierungen basieren auf dem Vererbungsprinzip, ein Objekt einer Regulierung kann vom einem Objekt einer höheren Regulierungsebene übersteuert werden. Anhand des Überlagerungsschlüssels werden die zu überlagernden Objekte identifiziert:

- *Case*: Name (Schlüssel)
- *Case Field*: Name
- *Case Relation:* Case Namen von Quelle und Ziel

- *Collector:* Name
- *Wage Type*: Lohnartnummer
- *Report:* Name
- *Report Parameter:* Name
- *Report Template:* Name
- *Lookup*: Name
- *Lookup Value:* Key

2.4.4 Case und Case Fields

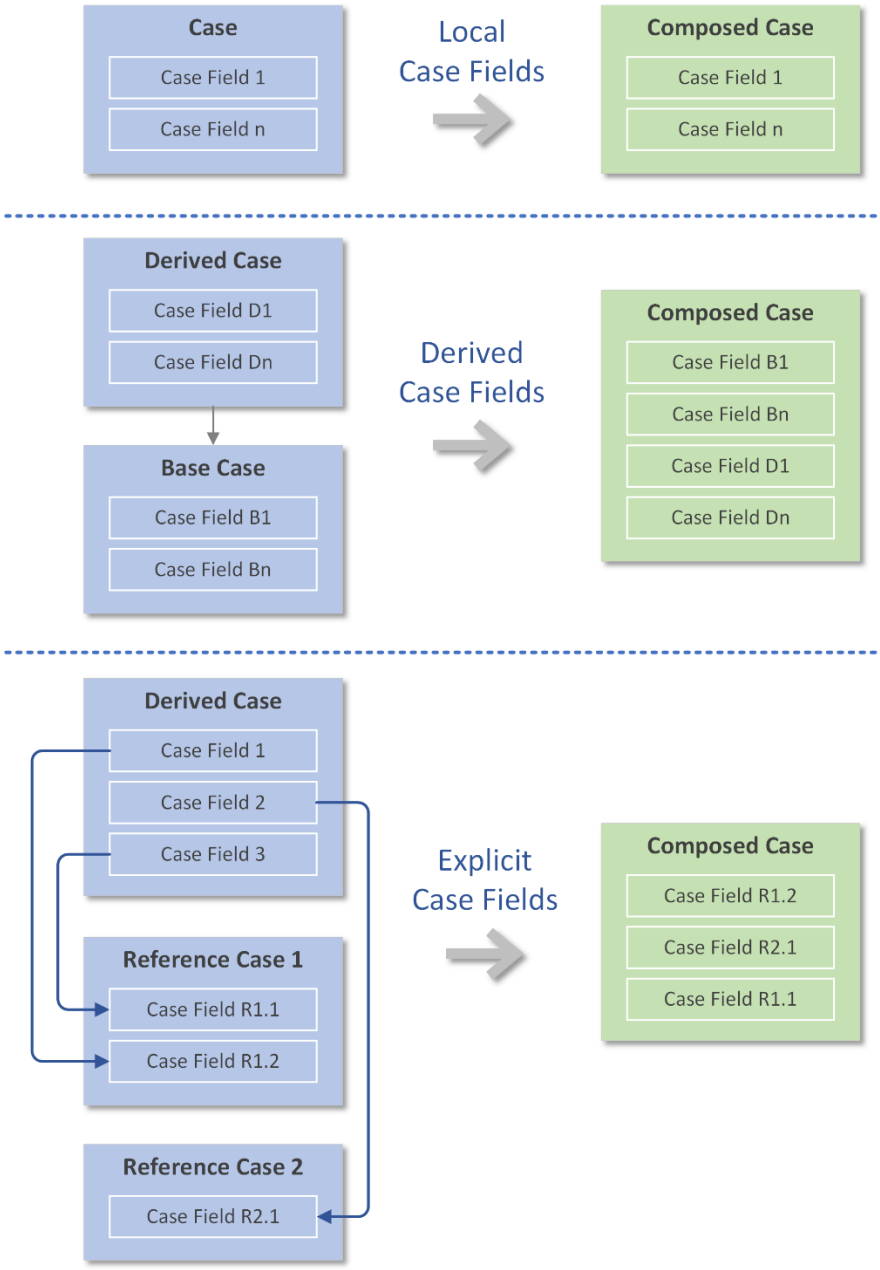
Das Case Objekt dient zur Erfassung von Falldaten. Das Case Field Objekt repräsentiert ein Eingabefeld im Case und das Case Relation Objekt bestimmt das Verhältnis (Verfügbarkeit und Werteübernahme) zwischen zwei Cases.

Bei der Erfassung des Fallwertes wird dessen Gültigkeitszeitraum mit dem von/bis Zeitpunkt bestimmt. Dadurch lassen sich komplexe Geschäftsfälle wie das Stornieren früherer Mutationen oder der Terminierung einer befristeten Lohnanpassung abbilden (siehe Kapitel *Zeittypen*). Dieses Merkmal unterscheidet die Payroll Engine wesentlich von konventionellen Softwarelösungen, welche die Daten in der Regel ersetzen.

Die Engine bietet verschiedene Ansätze für die Zusammenstellung von Cases und deren Case Fields:

- Deklaration der Case Fields im Case als integraler Bestandteil
- Vererbung der Case Fields von einem anderen Case
- Integration von spezifischen Case Fields aus verschiedenen Cases

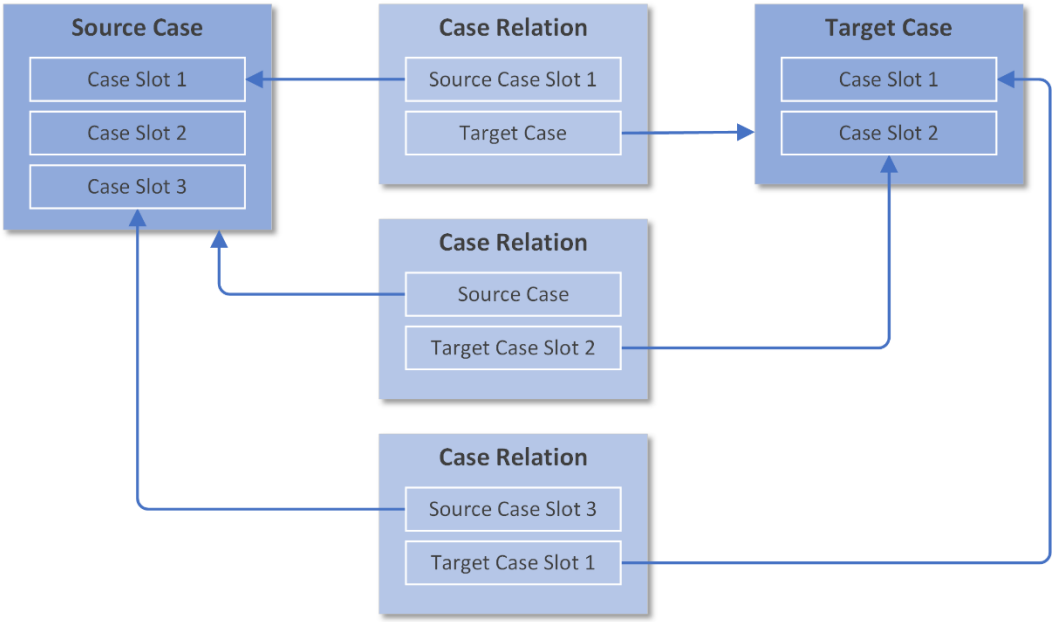
Die folgende Übersicht zeigt die möglichen Zusammensetzungen von Fällen, welche auch kombinierbar sind:



2.4.5 Case Slots

Um von einem Case mehrere Datensätze zu speichern (z.B. Kind oder Adresse), wird jede Variante einem Case Slot zugeordnet, welcher einen eindeutigen Namen besitzt.

Bei Beziehungen zwischen zwei Fällen lässt für Quelle und Ziel ein Case Slot bestimmen:



2.4.6 Lookups

Das Lookup beinhaltet Auswahldaten einer Regulierung für

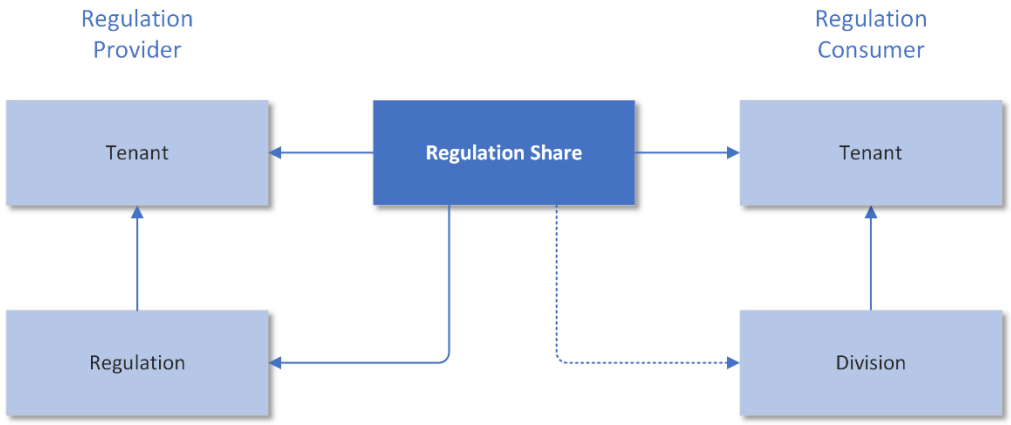
- fremdbestimmte Daten von externen Stellen wie z.B. eine Steuertabelle
- unveränderbare Auswahlwerte ohne Änderung während einer Lohnperiode
- Werte die per Regulierungsversion ändern können

Ein Lookup besteht aus einer Liste von Werten (Lookup Value), welche aus einem Schlüssel und einem Datenobjekt bestehen. Der Schlüssel ist ein Text, welcher innerhalb der Lookup-Werte eines Lookups eindeutig ist. Das Datenobjekt wird durch eine JSON-Struktur beschrieben. Es liegt in der Verantwortung des Nutzers, dass die JSON-Struktur bei allen Lookup Values identisch ist.

Neben der Suche eines Lookup Values anhand des Schlüssels, kann der Lookup Value anhand einer Wertigkeit (Dezimalzahl) gesucht werden. Dies kann genutzt werden, um ein Ergebnis für einen Wertebereich zu bestimmen.

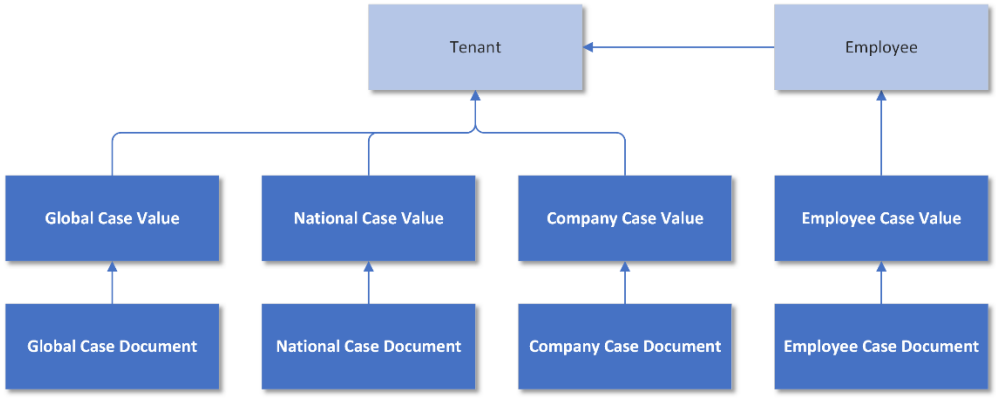
2.5 Geteilte Regulierungen

Die Engine bietet die Möglichkeit eine Regulierung für spezifischen Mandanten freizugeben. Dies ermöglicht dem Payroll Provider die zentralisierte Administration von Landes- und Branchenregulierungen. Durch Berechtigungen wird bestimmt, welche Regulierung für welchen Mandanten (wahlweise Division) verfügbar ist:



2.6 Falldaten

Für jedes Feld eines Falles (Case Field) kann ein Wert (Case Value) mittels Fallwechsel (Case Change) erfasst werden. Die Daten der Fälle werden physikalisch getrennt und nach National, 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 Daten des Unternehmens.
Company Case Document	Die Unternehmensdokumente.
Employee Case Value	Die Daten des Mitarbeiters.

Employee Case Document	Die Mitarbeiterdokumente.

Ein Case Value steht wahlweise für alle oder einen Unternehmensbereich (Division) zur Verfügung. Die Definition erfolgt im Datenfeld (Case Field) der Regulierung.

Falldaten sind unveränderbar und werden durch Fallwechsel erweitert. Basierend auf dieser Änderungshistorie können die Daten einer Lohnperiode bestimmt werden.

Der Wert eines Falles (Case Value) kann auf einen Gültigkeitszeitraum (Start/Ende) eingegrenzt werden (siehe Kapitel *Zeittypen*). Der Lohnlauf berücksichtigt den Wert nur, wenn dessen Zeitraum mit der Lohnperiode korrespondiert.

Für Fälle gelten folgende Typen:

- Mitarbeiter
- Unternehmen
- National
- Global

Die Falltypen können wie folgt in Regulierungen genutzt werden:

Regulierung	Global	National	Unternehmen	Mitarbeiter
Mandant	✓	✓	✓	✓
Business	✓	✓	✓	✓
National	✓	✓		
Global	✓			

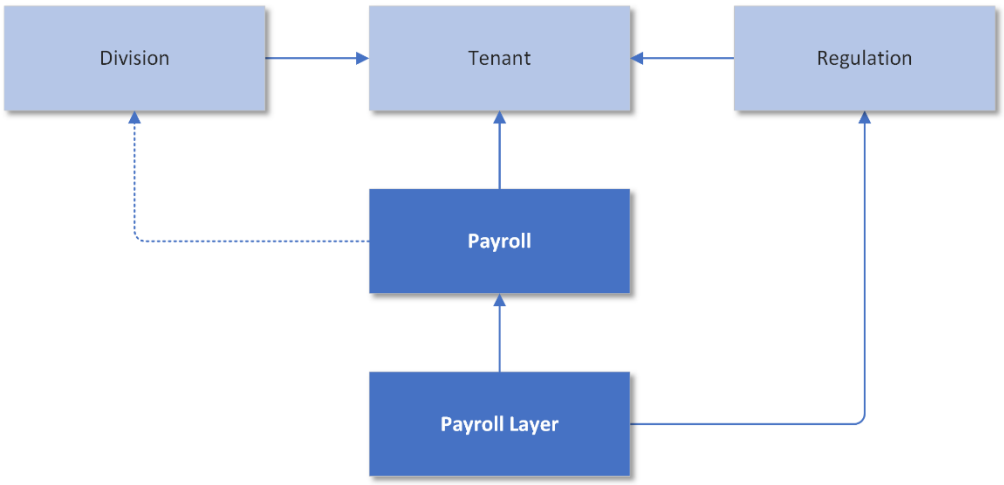
2.6.1 Falldaten stornieren

Ein Fall welcher für Stornierungen zugelassen ist, steuert das Rücksetzverhalten der Fallwerte anhand seines Stornotyps:

- Vorheriger Wert
- Wert initialisieren
- Wert invertieren
- Wert beibehalten

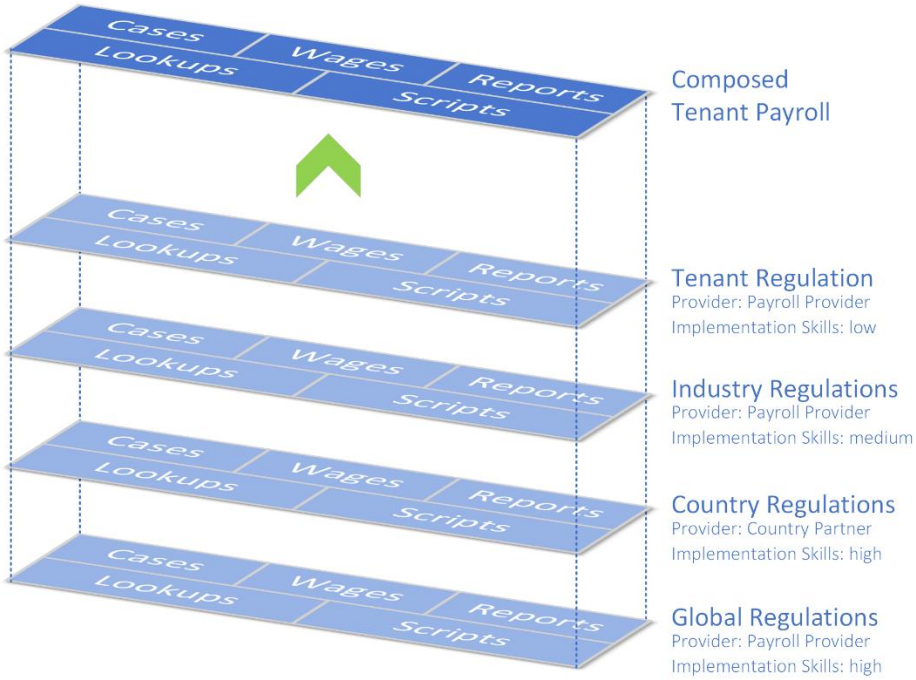
2.7 Payroll

Das Payroll Objekt verbindet mehrere Regulierungen in einem Schichtenmodell zu einem virtuellen Lohnmodell. Folgende Objekte sind involviert:



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

Die Payroll bietet die konsolidierte Sicht auf die Regulierung. Beispiel:



2.7.1 Payroll Schichten

Die die Auswertungsreihenfolge der Regulierungen werden im *Payroll Layer* anhand des Levels (1. Sortierkriterium) und der Priorität (2. Sortierkriterium). Die Anzahl der Schichten ist dabei unbegrenzt.
Das folgenden Beispiel zeigt drei Schichten und wie sich die Regulierungen zu einer dynamischen Payroll verbinden:

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 vererbt werden. Dabei werden die Werte aller Schichten dynamisch zu einem Objekt konsolidiert:

- Lokalisierungen
- Attribute
- Clusters
- Case Slots, Lookups und Actions
- Textfelder

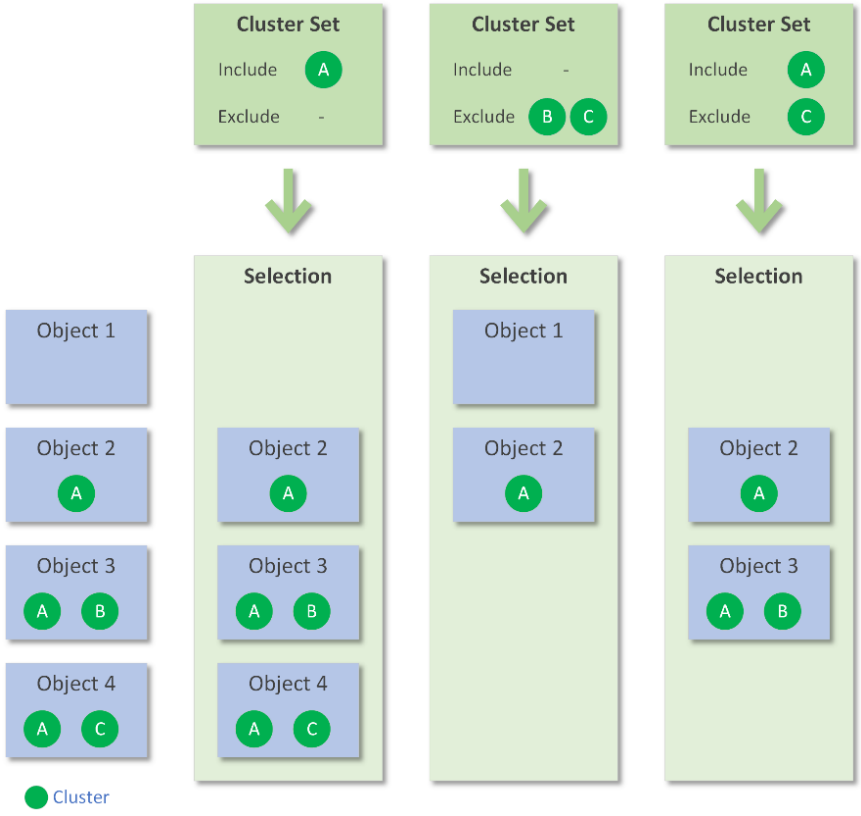
2.7.3 Payroll Clusters

Mittels Cluster können Regulierungsobjekte nach freien Kriterien gruppiert werden. Dies kann genutzt werden um:

- API-Abfragen zu filtern z.B. nur die Onboarding Fälle
- die Payroll Resultate einzugrenzen
- Regulierungsobjekte ein- oder auszuschliessen
- eine Regulierung abzugrenzen

Jedem Clusterobjekt können mehrere Clusternamen (analog Tags) zugeordnet werden. Die Abfrage der Objekte erfolgt über das *ClusterSet*, welches die zu inkludierten (Whitelist) oder die zu exkludierten (Blacklist) Clusters beinhaltet. Durch das Kombinieren von Whitelist-Cluster mit Blacklist-Cluster können inkludierte Clusters weiter eingegrenzt werden.

Die folgende Abbildung zeigt verschiedene Szenarien zur Auswahl von Clusterobjekten:



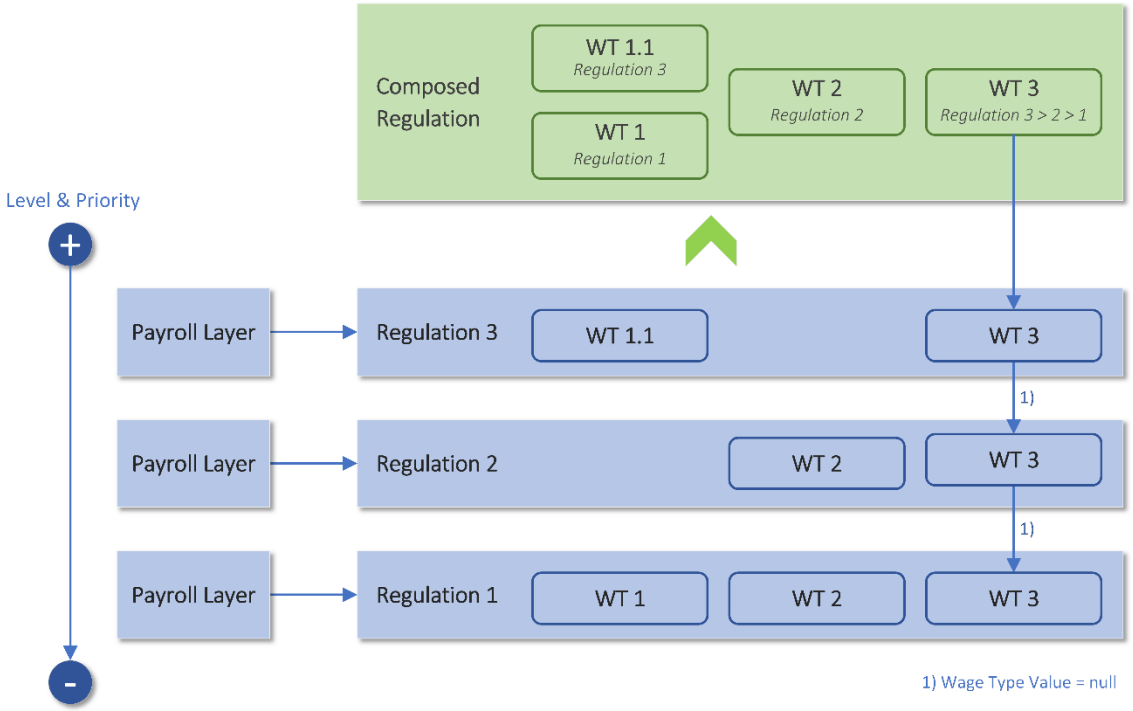
Cluster Sets sind im Objekt *Payroll* definiert und können in den REST Abfragen der Regulierungsobjekte genutzt werden. Folgende Cluster Sets bestehen:

- *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

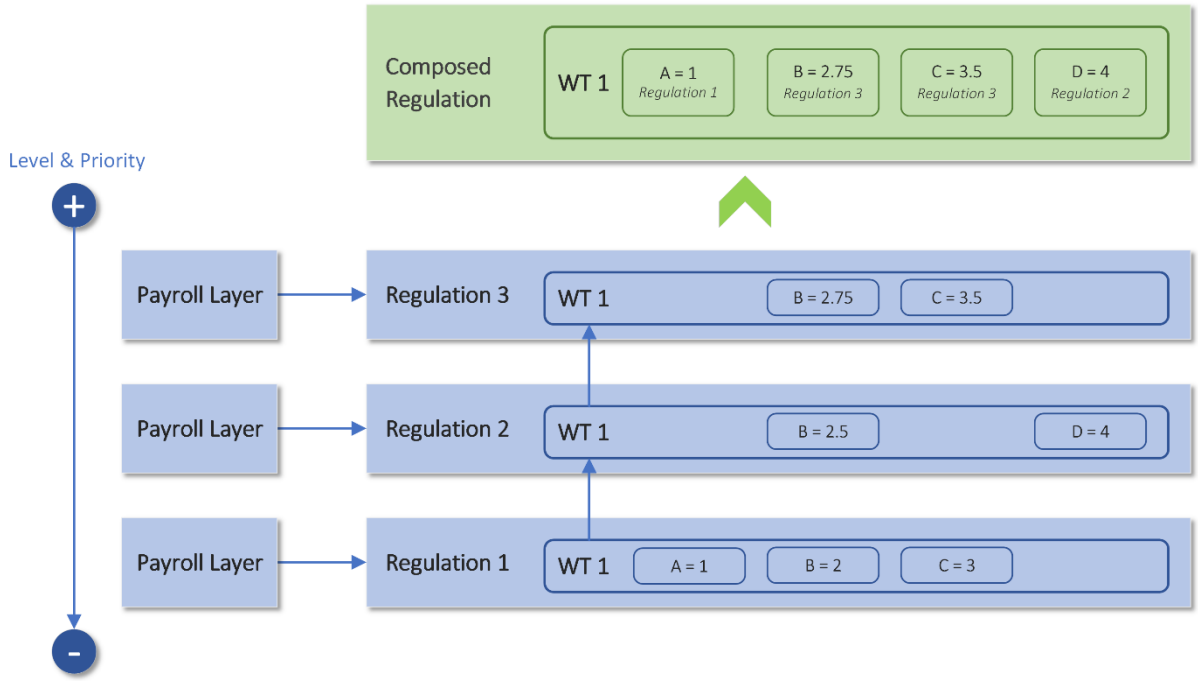
Die Scripting API ermöglicht für spezifische Domainobjekte das Laufzeitverhalten zu steuern (siehe Kapitel *Scripting API*). Die Auswertung der Scripts erfolgt anhand der Auswertungsreihenfolge im Payroll Layer.

Das folgende Beispiel zeigt die Berechnung von Lohnarten:



Durch die Rückgabe eines undefinierten Rückgabewerts (*null*) einer Scripting Funktion, im Beispiel oben WT2, wird die Verarbeitung an die unterliegende Regulierung delegiert.

Im Gegensatz dazu werden Objektattribute in der entgegengesetzten Richtung zusammengeführt, so dass der Wert von einer überlagerten Regulierung überschrieben werden kann:



2.8 Lohndatenberechnung

2.8.1 Lohnarten

Die Berechnung des Mitarbeiterlohnes erfolgt mittels Lohnarten, welche in der Reihenfolge ihrer Lohnartennummer berechnet werden. Zu jeder Lohnarten können ein oder mehrere Kollektoren zugeordnet werden, welche die fortlaufenden Lohnbasen berechnen. Die Anzahl der Lohnarten ist nicht limitiert und ergibt sich aus der Summer der Payroll-Regulierungen.

2.8.2 Kollektoren

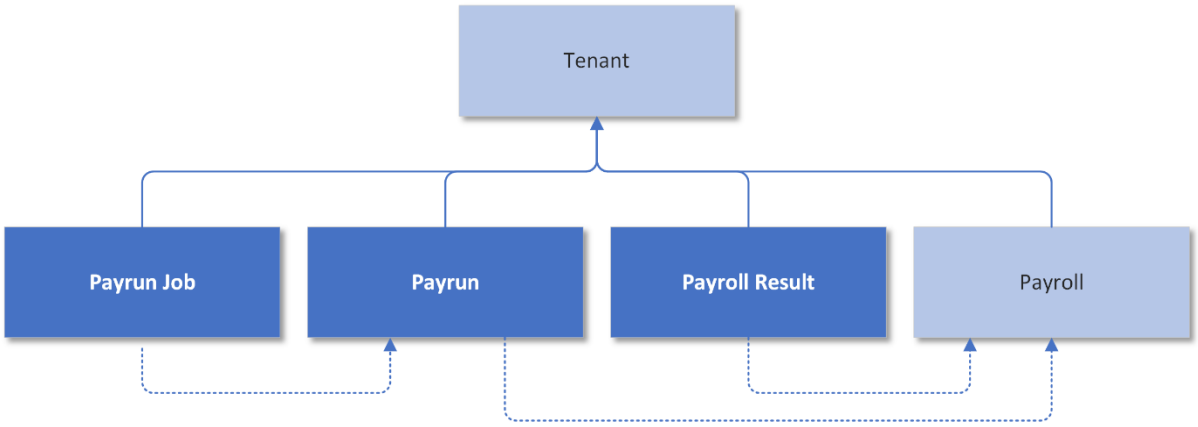
Kollektoren tragen die Werte der Lohnarten zusammen und bieten verschiedene vordefinierte Operatoren:

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

Neben den vordefinierten Typen von Aggregierungen, lässt sich der Wert mit der Scripting Funktion *Collector Apply* individuell bestimmen.

2.9 Lohnlauf

Die Berechnung der Lohndaten erfolgt mit folgenden Objekten:



Payrun Job	Die Steuerung des Lohnlaufes.
Payrun	Der Lohnlauf basierend auf einem Payroll.
Payroll Result	Die Ergebnisse des Lohnlaufes.

Die Lohnlauf Sequenz ist im Kapitel *Regulation Scripting* beschrieben.

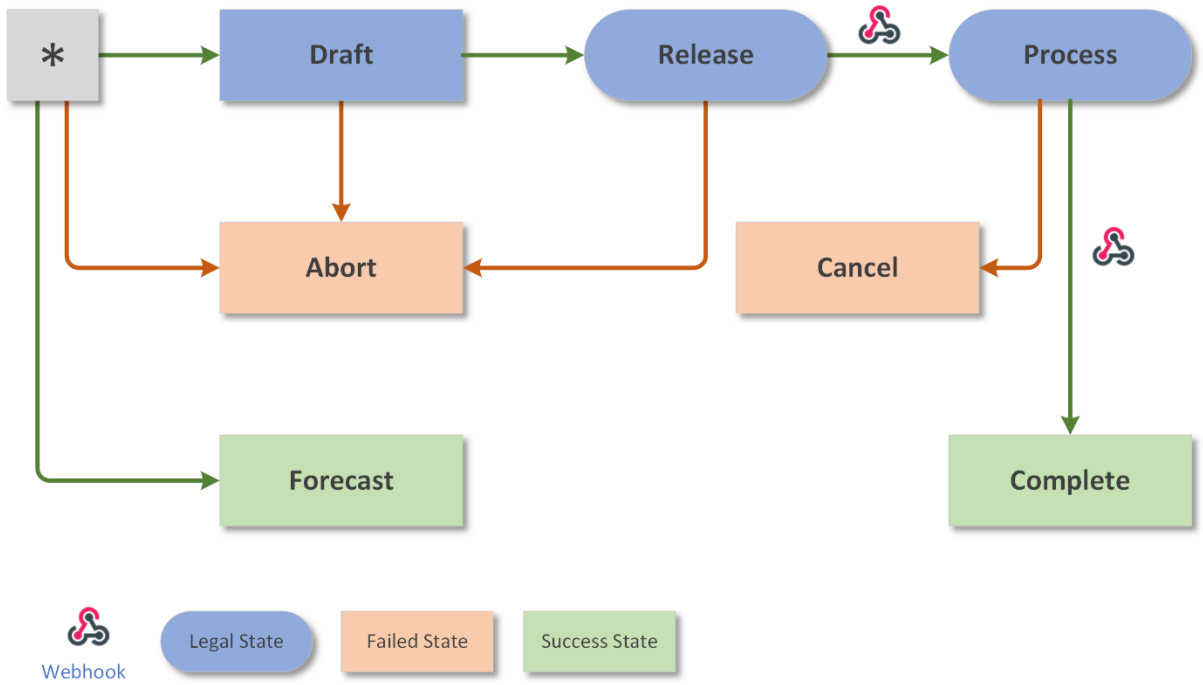
2.9.1 Lohnlauf Job

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

Der Payrun Job bestimmt für welchen Zweck die Ausführung erfolgt:

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

Die Steuerung des Payrun Jobs erfolgt mit dem Jobstatus:



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

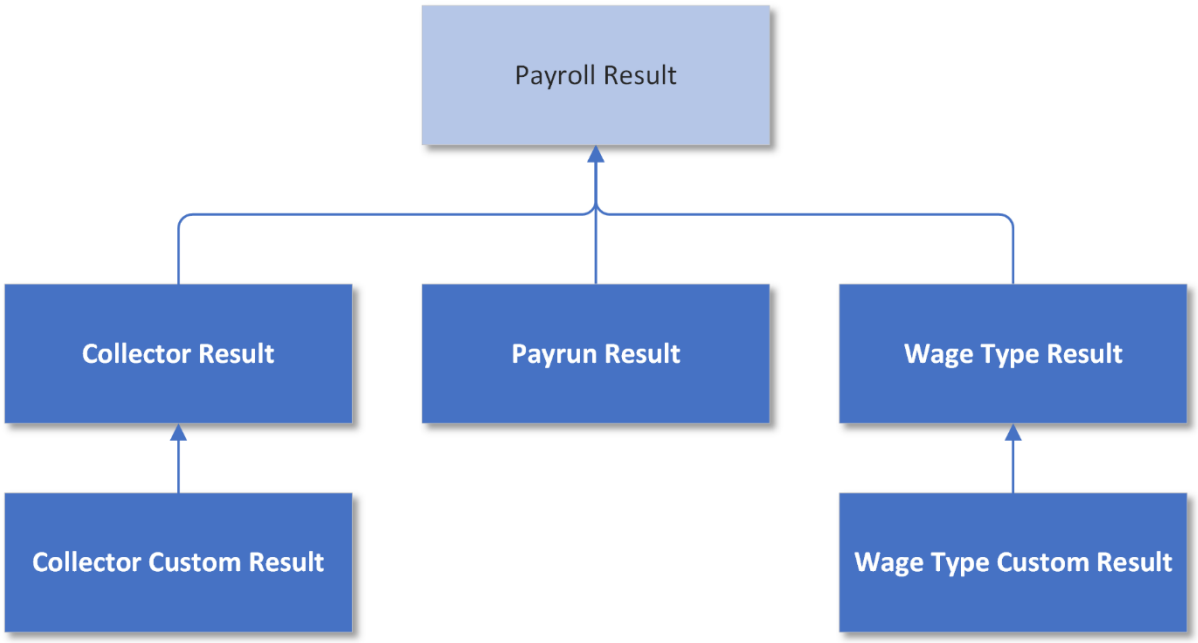
Pro Payroll kann nur ein Job im Status *Draft* existieren. Im Status *Release* und *Process* sind mehrere Jobs mit demselben Status möglich.

2.9.2 Lohnlauf Neustart

Im Lohnlauf werden alle Lohnarten in der Reihenfolge der Lohnartnummer abgearbeitet. Für Spezialfälle kann der Lohnlauf eines Mitarbeiters neu gestartet werden. Jede Lohnart hat einen Laufzähler und mittels *Payrun Laufzeitwerte* können Daten zwischen den Läufen ausgetauscht werden.

2.9.3 Payroll Resultate

Die Ergebnisse des Lohnlaufes werden in verschiedenen Objekten geführt:



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

In der Payroll wird mit Cluster Sets (siehe Kapitel *Objektvererbung*

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

- Lokalisierungen
- Attribute
- Clusters
- Case Slots, Lookups und Actions
- Textfelder

Payroll Clusters) gesteuert, welche Resultate zu erstellen sind.

2.10 Spezielle Lohnläufe

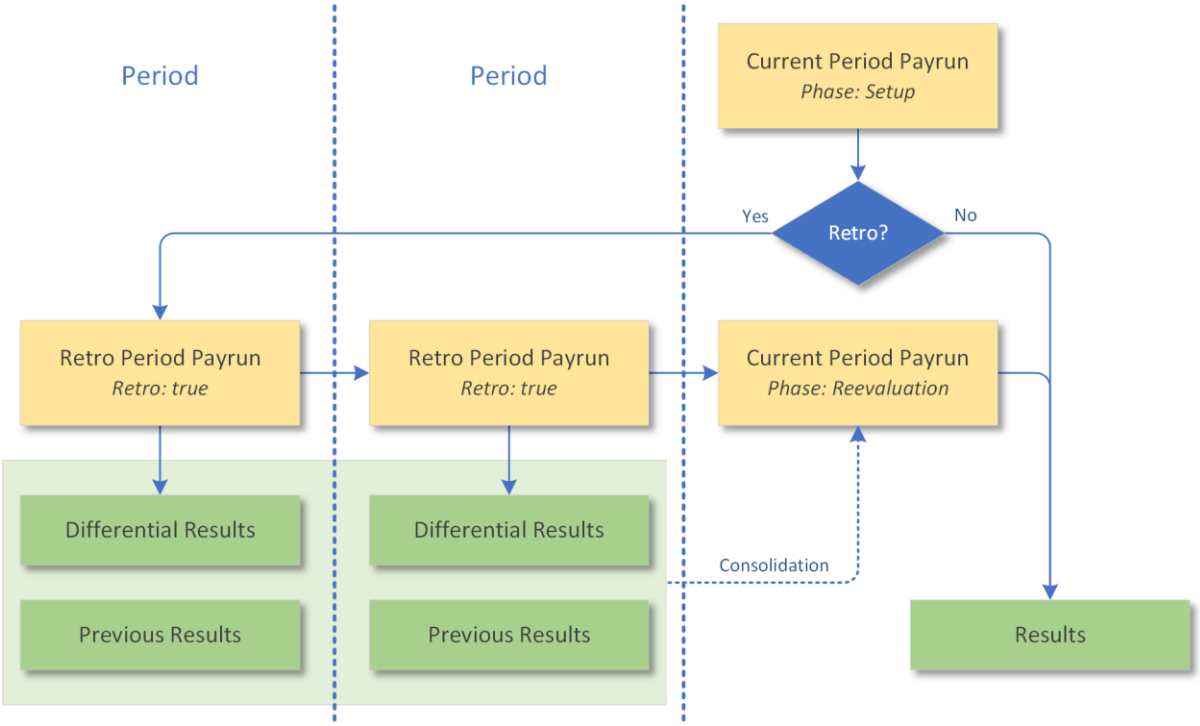
2.10.1 Inkrementeller Lohnlauf

Erfolgen innerhalb einer Periode mehrere Lohnläufe, werden mittels inkrementellen Payrun Jobs nur die, gegenüber den Vorläufen, geänderten Resultate erzeugt. Zur Ermittlung der aktuell gültigen (konsolidierten) Resultate einer Lohnperiode bietet die REST API entsprechende Endpunkte an.

2.10.2 Rückrechnung

Rückrechnungen entstehen automatisch, wenn seit dem letzten Lohnlauf neue Mutationen bestehen, welche sich auf Vorperioden auswirken.

Die Rückrechnung besteht aus folgender Sequenz:



Die Schritte in der Rückrechnung sind:

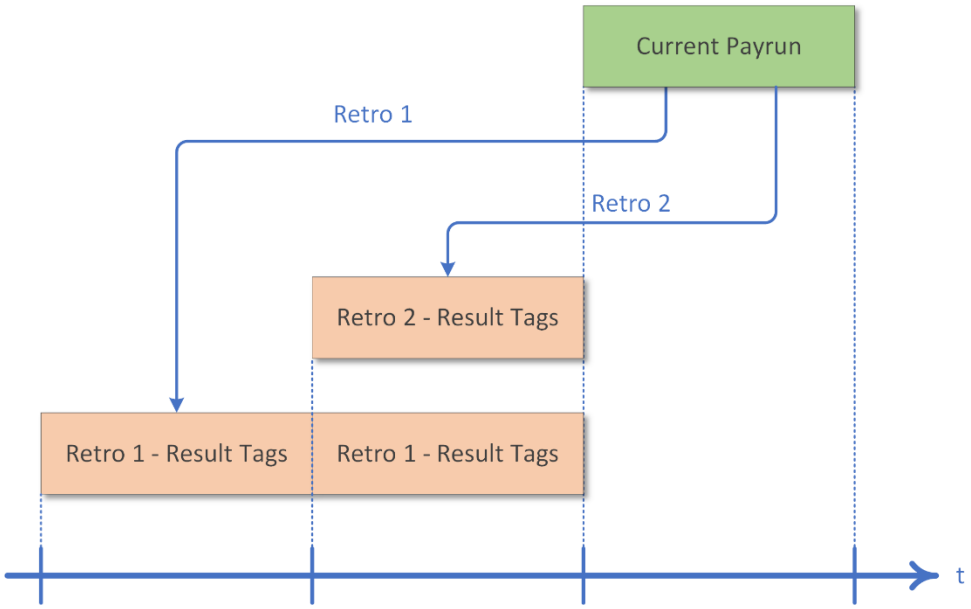
1. Berechnung der aktuellen Lohnperiode (mit Mutation auf Vorperiode)
   → Ausführungsphase *Setup*
   → Transiente Resultate
2. Berechnung aller Vorperioden startend bei der Mutationsperiode bis zur Vorperiode
   → Inkrementelle Resultate der Vorperioden speichern
3. Neuberechnung der aktuellen Lohnperiode
   → 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 unbeschränkt, wobei eine Einschränkung auf den aktuellen Lohnzyklus möglich ist.

2.10.3 Manuelle Rückrechnung

Zusätzlich zur automatisch ausgeführten Rückrechnung, kann mittels die Rückrechnungen manuell ausgelöst werden. Mittels Payroll Resultat-Tags werden die in der Rückrechnung generierte Ergebnisse markiert. Bei der Abfrage der Payroll Resultate können die Tags als Filterkriterien genutzt werden.

Im folgenden Szenario löst der Lohnlauf der aktuellen Periode zwei Rückrechnungsläufe aus:



2.10.4 Lohnlauf Neustart

In besonderen Fällen ist es erforderlich, den Lohnlauf neu zu starten. Anhand des Lohnlaufzählers kann die Lohnartenfunktion entsprechend reagieren (siehe *WageTypeFunction.ExecutionCount* und *WageTypeFunction.RestartExecution()*). Weitere Wiedereintritts-Informationen können in der Payrun-Runtime geführt werden.

2.11 Forecast

Zur Erstellung von Prognosen sind zwei Eingriffe erforderlich:

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

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

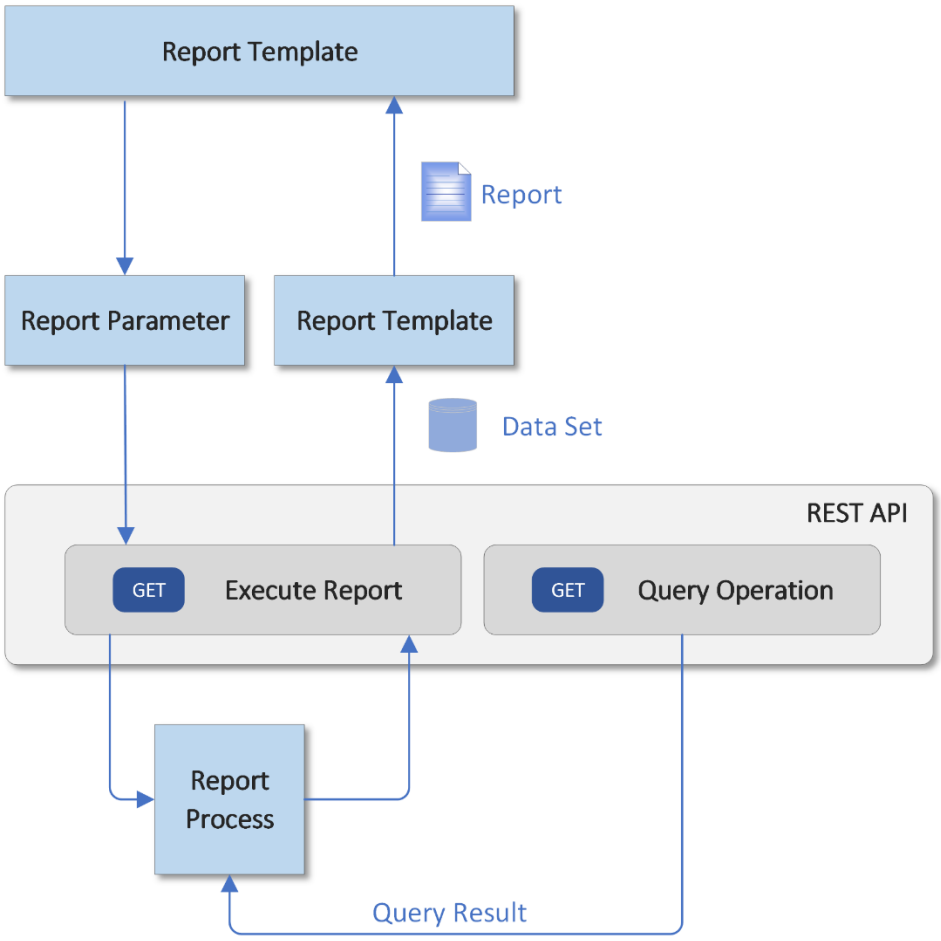
2.12 Report

Der Regulation Report bestimmt anhand von Reportparametern die Daten einer Auswertung. Die Generierung des Reports 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), welches vom Client in das Zieldokument transformiert wird. Zur Ermittlung der Reportdaten können alle GET Endpunkte der REST API als Datenquelle dienen.

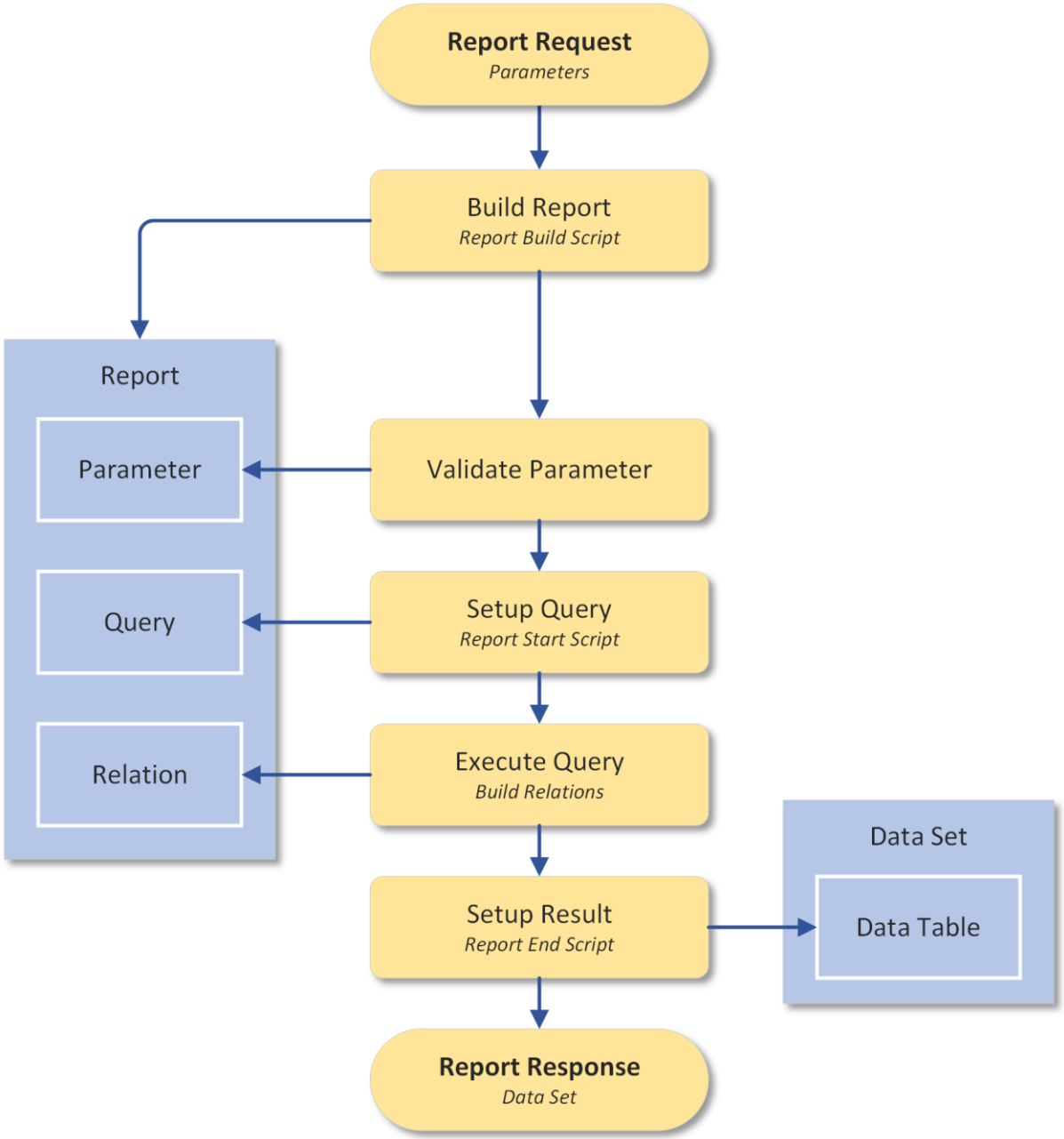
Report Generierung:



Das sprachbezogene Reportlayout (*ReportTemplate*) ist für die Berechnung der Reportdaten irrelevant und wird vom Client durch sein Umwandlungsverfahren bestimmt (*.docx*, *.rdlc* usw).

Zusätzlich besitzt das *ReportTemplate* ein Feld für ein Schema, welches der Client zur Validierung der Reportdaten (JSON/XML/usw.) nutzen kann.

Die Erstellung eines Reports erfolgt in folgenden Schritten:



1. Erstellung des Reports: Die Reportparameter im *Report Build Script* angepasst werden.
2. Überprüfung und Vervollständigung der Parameter
3. Aufbereitung der Abfragen (Queries): Die Abfragen sind im Report vorgegeben und können im *Report Start Script* angepasst werden.
4. Abfrage der Reportdaten: Der Report kann mehrere Abfragen ausführen, wobei die Daten eines API-Endpunktes (GET) in eine Tabelle (Memory) gespeichert werden. Abschliessend werden die Relationen zwischen den Tabellen gesetzt.
5. Nachbereitung der Reportdaten: Im *Report End Script* können die Tabellen dem gewünschten Zielformat angepasst werden. Zur Behandlung komplexer Fälle sind im Script Laufzeitabfragen möglich.

Jeder Schritt ist optional und kann je nach Reporttyp variieren. Ein einfacher Report kann lediglich aus Abfragen bestehen. Ein komplexer Report kann die gesamte Logik in der Nachbearbeitung abhandeln.

Weitere Reporting Features:

- Lokalisierte Texte
- Ausgabe benutzerdefinierte Attribute
- Nachführen der Dokument Metadaten inklusive Abfrageparameter
- Umfangreiche Bearbeitungsmöglichkeiten der Reporttabellen im Scripting siehe https://docs.microsoft.com/en-us/dotnet/api/system.data.dataset
- Kategorisierung von Reports zur Eingrenzung der Auswahl im Client

3 Basiskonzepte

3.1 Sprachen

Der Payroll Dienst unterstützt Lokalisierungen von Texten in 48 Sprachen. Übersetzt wird der englische Vorgabewert, welcher das sprachneutrale Referenzieren zwischen Objekten ermöglicht.

3.2 Audit Trail

Für die zentralen Payroll Daten führt die Payroll API jede Änderung im Audit Trail. Dies gilt für alle Regulierungsobjekte und Falldaten. Der Audit-Trail der Falldaten dient auch als Grundlage zur Berechnung der Zeitraumwerte.

3.3 Datentypen

Der Payroll Dienst bietet folgende Datentypen an:

Datentyp	Basistyp (JSON)	C# Typ	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

In der Zuordnung zum Kalender orientiert sich der Payroll Dienst an folgenden Definitionen:

- Periode: die Lohnperiode (Monat, Woche, 14 Tage ...)
- Zyklus: der Wiederholungszyklus (Jahr, Halbjahr, Quartal …)

3.4.1 Weltzeit

Alle Datumswerte werden in UTC geführt, andere Zeitmodelle führen zu einem Fehler.

3.4.2 Periodenberechnung

Die Berechnungsgrundlage der Periode wird durch den Berechnungsmodus bestimmt, welcher im Payroll bestimmt und im Wage Type übersteuert werden kann. Derzeit stehen folgende Berechnungsmodi zur Verfügung:

- Monat Kalendertage: Anzahl Tage im Monat
- Monat Durchschnitt Kalendertage: Jahresdurchschnitt (z.B. 30)
- Monat Werktage: Tage basierend auf den Arbeitstagen der Woche
- Monat Durchschnitt Werktage: Jahresdurchschnitt (z.B. 21.75)
- Woche Werktage: Tage basierend auf den konfigurierten Arbeitstagen der Woche

3.4.3 Zeittypen

Die Berechnung der Zeitwerte basiert auf folgenden Zeittypen:

Typ	Beschreibung	Periode Start	Periode Ende	Perioden-werte
Timeless	Wert ohne Zeitbindung (Vorsorgepflicht).	-	-	1
Moment	Wert welcher einem Zeitpunkt zugeordnet ist (Bonus).	x	-	1..n Summe
Period	Wert für einen Zeitraum (Arbeitsstunden).	x	(x)	n überlagernd
ScaledPeriod	Wert wird auf die Kalenderperiode tageweise aufgeteilt (Monatslohn).	x	(x)	n überlagernd

3.4.4 Zeitraumwerte

Die Bestimmung der Fallwerte ist mit folgenden Zeiträumen möglich:

- Aktuelle, vergangene oder zukünftige Perioden (z.B. Monat)
- Aktuelle, vergangene oder zukünftige Zyklen (z.B. Jahr)
- Benutzerdefinierte Periodenzeitraum

3.4.5 Zeitraumwerte für Mutationen

Bei der Berechnung eines Zeitraumwertes werden alle relevanten Mutationen innerhalb des Periodenzeitraums gemäss dem Kalender anteilsmässig aufgeteilt (z.B. untermonatige Lohnanpassung).

Die Scripting API bietet die Möglichkeit, die Aufteilungslogik des Kalenders zu nutzen, was die Berechnungssyntax massiv vereinfacht. Zwei Fallwerte mit verschiedenen Mutationen können mit den gewohnten mathematischen Operatoren (Plus, Minus, Multiplikation und Subtraktion) berechnet werden.

3.4.6 Kalenderkonfiguration

Die Zeitberechnungen der Payroll Engine basieren auf folgender Kalenderkonfiguration:

- Erster Monat im Jahr (Systemdefault: *January*)
- Regel für die erste Woche im Jahr CalendarWeekRule.FirstFourDayWeek
- Erster Tag in der Woche (*Monday*)
- Die durchschnittliche Anzahl der Monatstage (*30*)
- Die durchschnittliche Anzahl der Arbeitstage (*21.75*)
- Die Arbeitstage (*Montag bis Freitag*)
- Die Kalender Berechnungsmethode (*CalendarCalculationMode.MonthCalendarDay*)

Die Kalenderkonfiguration gilt für den Mandanten und/oder dem Lohnlauf und wird nach folgender Priorität aufgelöst:

1. Division Kalender (Quelle: *Division.Calendar*)
2. Mandant Kalender (*Tenant.Calendar*)
3. Default Kalender (*CalendarConfiguration.DefaultConfiguration*)
4. System Kalender (unveränderbar, siehe oben)

3.5 Objekteigenschaften

Jedes API-Objekt besitzt folgende Grundeigenschaften:

- Identifikation (Fortlaufende Nummer)
- Erstellungsdatum in UTC (*Created*)
- Datum der letzten Änderung in UTC (*Updated*)
- Objektstatus (Aktiv, Inaktiv)

Das Erstellungs- und Änderungsdatum wird von System geführt und ist grundsätzlich nicht änderbar. Weil das Erstellungsdatum des Objektes bei der Berechnung der Zeitraumwerte relevant ist, bietet die Provider API entsprechende Patch-Methoden, um die Datumswerte des Objektes zu modifizieren. Dieses Verhalten ist bei Integrationen und Migrationen erforderlich.

Neue erstellte Objekte sind aktiv und können mittels Objektstatus nachträglich deaktiviert werden. Inaktive Objekte werden im Payroll sowie im Payrun ignoriert.

3.6 Objektattribute

Objektattribute sind benutzerdefinierte Daten, deren Inhalt keine Auswirkung auf das Verhalten des Payroll Dienstes hat. Attribute können für folgende Szenarien genutzt werden:

- Speicherung von Zusatzdaten (falls kein Zeitwert oder Fallwert)
- Eingabeattribut für Payroll Clients (*Case Field*)
- Steuerung/Parametrisierung von Scripting Funktionen (*Collector* oder *Wage Type*)
- Steuerung des Lohnlaufes (*PayrunJob*)
- Speicherung zusätzlicher Resultate mit z.B. Meldedaten (*Wage Type Result*)

Objekteattribute werden in einer Werteliste (Dictionary) geführt, wobei gewisse Objekte API-Endpunkte anbieten, um die Attribute direkt zu behandeln. Attribute können in Abfragen verwendet werden (sieh Kapitel *Listenabfragen*):

Übersicht der API-Objekte mit Attributen:

Attributobjekt	Attribut API	Audit
Tenant	x	
User	x	
Task	x	
Division	x	
Employee	x	
Payroll	x	
Payroll Layer	x	
Payrun Job	x	
Webhook	x	
Regulation	x	
Case		x
Case Field		x

Attributobjekt	Attribut API	Audit
Case Relation		x
Collector		x
Wage Type		x
Report	x	x
Case Value		
Payroll Result		

4 Payroll Testing

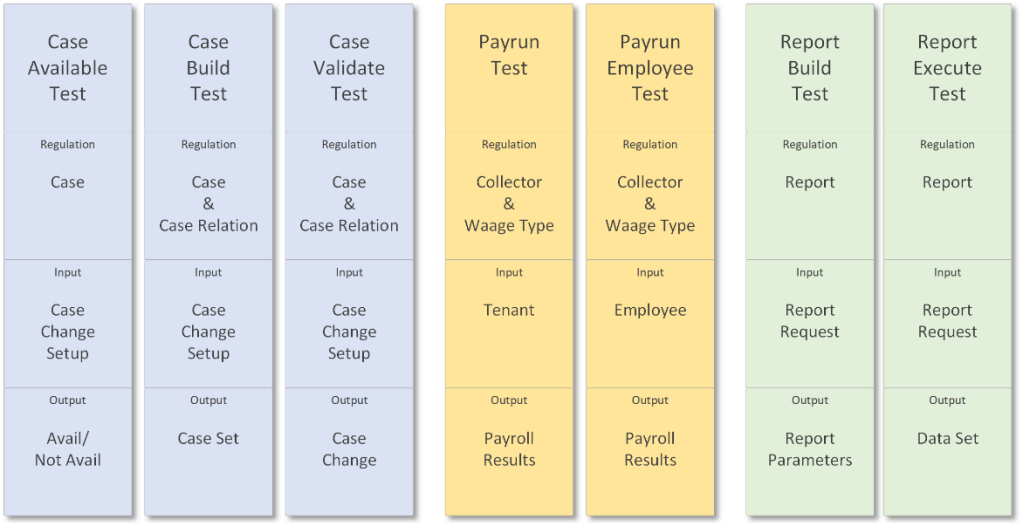
Basierend auf modernen Ansätzen in der Softwareentwicklung, bietet die Payroll Engine auch die testgetriebene Entwicklung des Lohnes an. Mittels Client Services Tools können folgende Aspekte im Lohnwesen automatisiert getestet werden:

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

Das automatisierte Testen in drei Bereichen möglich:

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

All Regulierungsobjekte lassen sich mit folgenden Tests automatisiert überprüfen:



4.1 Fall Tests

Mit den Fall Tests kann die Verfügbarkeit, Generierung sowie die Validierung eines Geschäftsfalles getestet werden. Die Fall Tests können durch Konfigurierung (JSON) oder durch Programmierung (C# mit den Client Services) bestimmt werden.

4.2 Lohnlauf Tests

4.2.1 Mandantenlohnlauf Tests

Im Mandantenlohnlauf Test wird ein neuer Mandant erstellt und die Ergebnisse der Lohnläufe mit den erwarteten Ergebnissen verglichen. Der Mandant wird nach dem Test entfernt.

4.2.2 Mitarbeiterlohnlauf Tests

Für umfangreichere Mandantenszenarien sind Tests auf einzelne Mitarbeiter möglich. Der Mitarbeiterlohnlauf-Test setzt einen bestehenden Mandanten voraus. Der Lohnlauf erfolgt auf der Kopie eines bestehenden Mitarbeiters. Die Bereinigung der Testdaten erfolgt manuell.

4.3 Report Tests

Für Payroll Reports kann die Erstellung des Reports (Report Parameter), sowie die Ausführung der Reports getestet werden. Die Report Tests können durch Konfigurierung (JSON) oder durch Programmierung (C# mit den Client Services) bestimmt werden.

4.4 Tests anwenden

Der Test wird in JSON definiert und mit der Payroll Konsolenanwendung (siehe Kapitel *Payroll Konsole*) ausgeführt. Die Testergebnisse werden am Bildschirm angezeigt und Testfehler in Logdateien protokolliert.

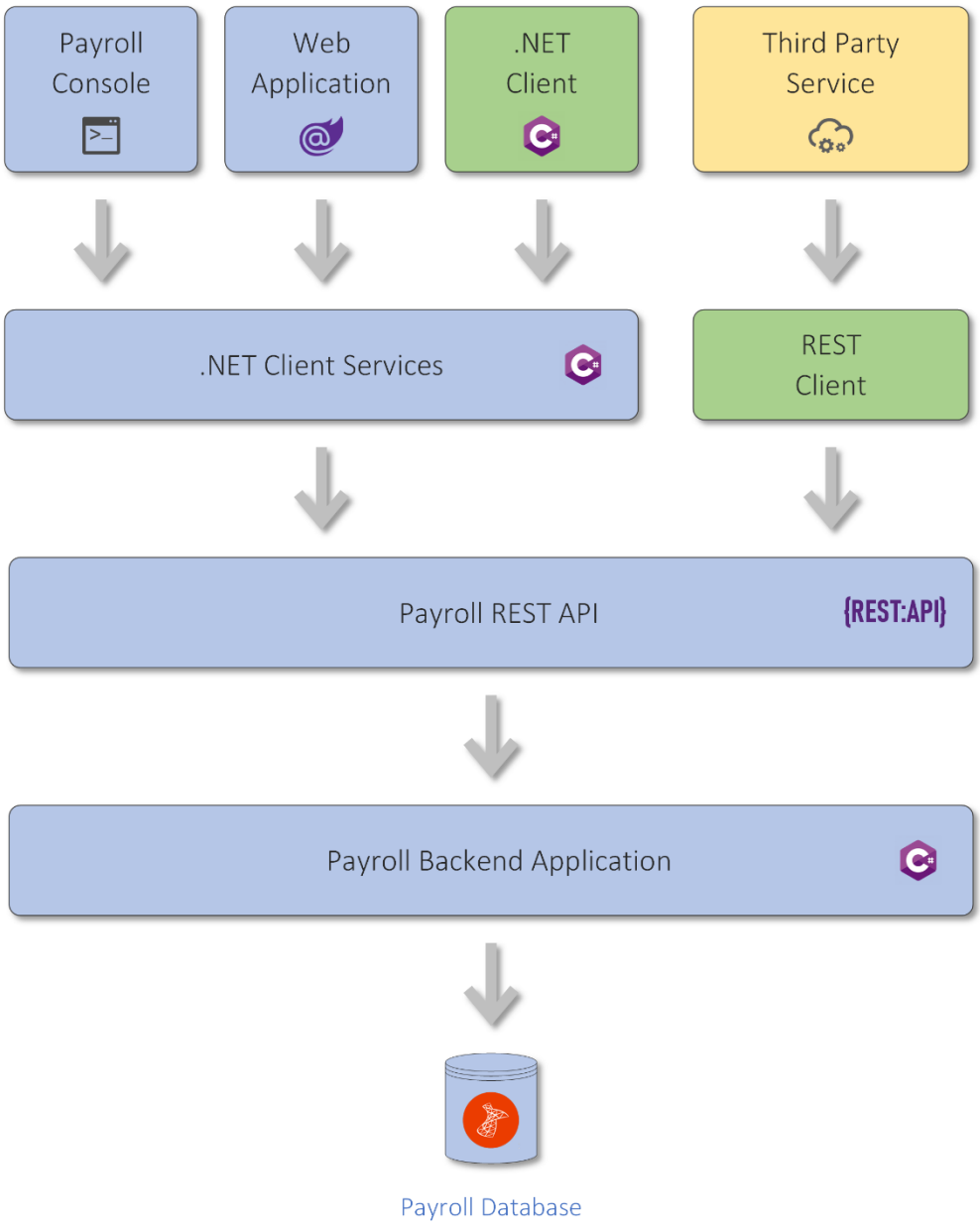
Die Testfunktionen stehen auch als Programmiermodule zur Verfügung (siehe Kapitel *Scripting API*) und können individuell erweitert werden.

5 Payroll Backend

Die Payroll Backend setzt sich aus folgenden REST APIs zusammen:

- Provider API: Systemanbieter
- Onboarding API: Payroll Dienstleister
- Human Resources API: Personalwesen
- Mitarbeiter API: Mitarbeiter (Zukunftsszenario)

Die APIs bauen hierarchisch aufeinander auf, wodurch die Provider API den vollen Funktionsumfang abdeckt:

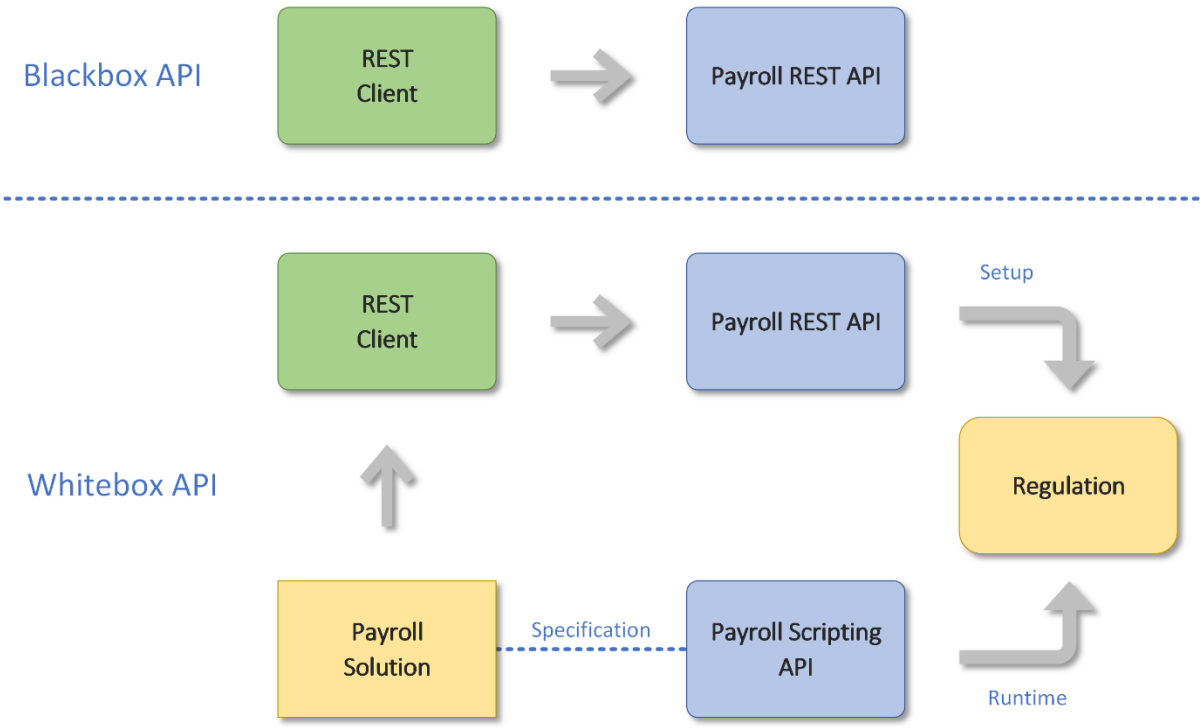


Die REST Endpunkte sind im Dokument *PayrollRestServiceEndpoints.xlsx* beschrieben.

5.1 Whitebox API

Die Payroll Engine ist ein API-First Produkt, welches sich von den herkömmlichen APIs in der Nutzungsweise unterscheidet. Die meisten APIs sind aus Sicht vom Business Blackboxen, das heisst die Funktionsweise ist nicht bekannt und kann nicht wesentlich beeinflusst werden.

Die Engine besitzt eine offene Architektur als eine Whitebox REST API, welche mittels zusätzlicher Schnittstelle, die Scripting API, die Adaptierung zum Business ermöglicht. Die folgende Abbildung zeigt die Differenzen zwischen Blackbox und Whitebox REST APIs auf:



5.2 Technologien

Im Payroll Backend werden folgenden Technologien verwendet:

- REST API — OpenAPI 3.0 (OAS3)
Media type: application/json
- Scripting API — C# 11
- Server-Applikation — .NET 7
- Datenbank — SQL-Server 2019

5.3 Listenabfragen

Zur Einschränkung von Listen lassen sich API Abfragen nach folgenden Kriterien einschränken:

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

Als Abfrageergebnisse sind die Objektliste, die Anzahl der Listeneinträge oder beides möglich.

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

- Text Attribut: *TA_*
  → Beispiel: *TA_MyName*
- Datum Attribut: D*A_*
  → Beispiel: *DA_ProjectCreated*
- Zahl Attribut: *NA_*
  → Beispiel: *NA_SecurityLimit*

5.4 Webhooks

Der Payroll Dienst bietet Webhooks für folgende Aktionen:

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	Legal Payrun Job freigegeben zur Verarbeitung.	*Request: PayrunJob*	✓
PayrunJobCompleted	Legal Payrun 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* (Schema siehe REST API). Von der Webhook Antwort wird der http Statuscode sowie der Content (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 zur Systemadministration:

- Die Applikation beenden
- Den Applikations-Cache bereinigen
- Ermittlung der Verfügbaren Report-Endpunkte ermitteln (siehe Kapitel *Report*)

6 Scripting API

Die Scripting API ermöglicht dem Nutzer das Laufzeitverhalten in folgenden Bereichen zu beeinflussen:

Case Management

- Verfügbarkeit eines Falles bestimmen
- Aufbau eines Falles (Werte und Bezugsfälle) bestimmen
- Verhalten zwischen Fällen bestimmen (Relationen)
- Falldaten validieren
- Neuen Fall bestätigen

Lohnlauf

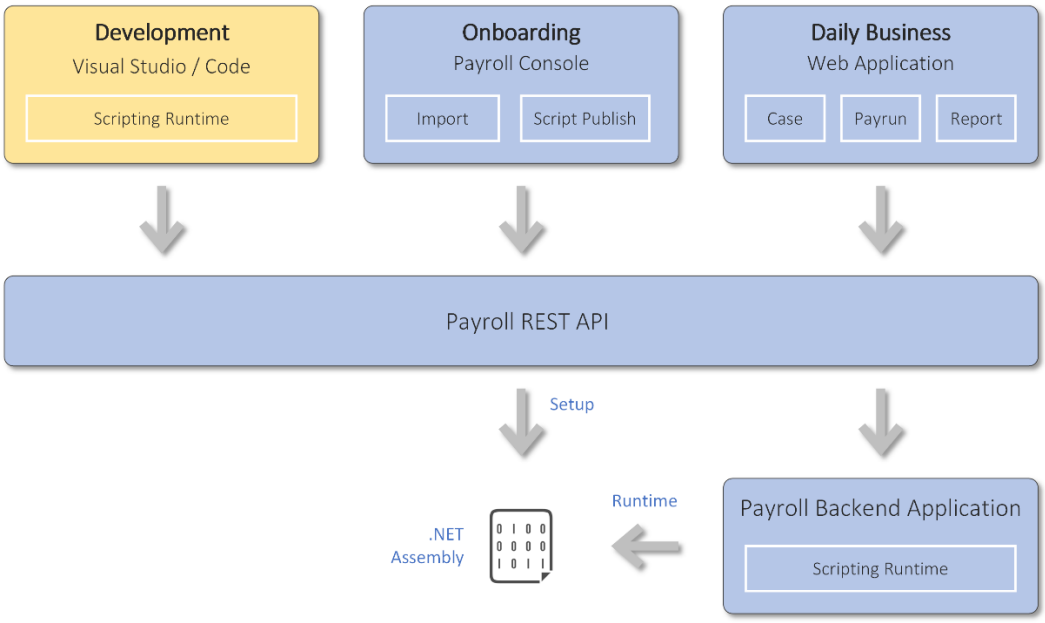
- Mitarbeiter und Lohnarten eingrenzen
- Lohn mit Kollektoren und Lohnarten berechnen und speichern
- Mit externen Diensten kommunizieren (Webhooks)
- Zusatzdaten berechnen und speichern

Report

- Reportparameter bestimmen
- Reportdaten berechnen

Neben den Standard Webhooks (siehe Kapitel *Webhooks*) sind benutzerdefinierte Webhooks möglich.

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

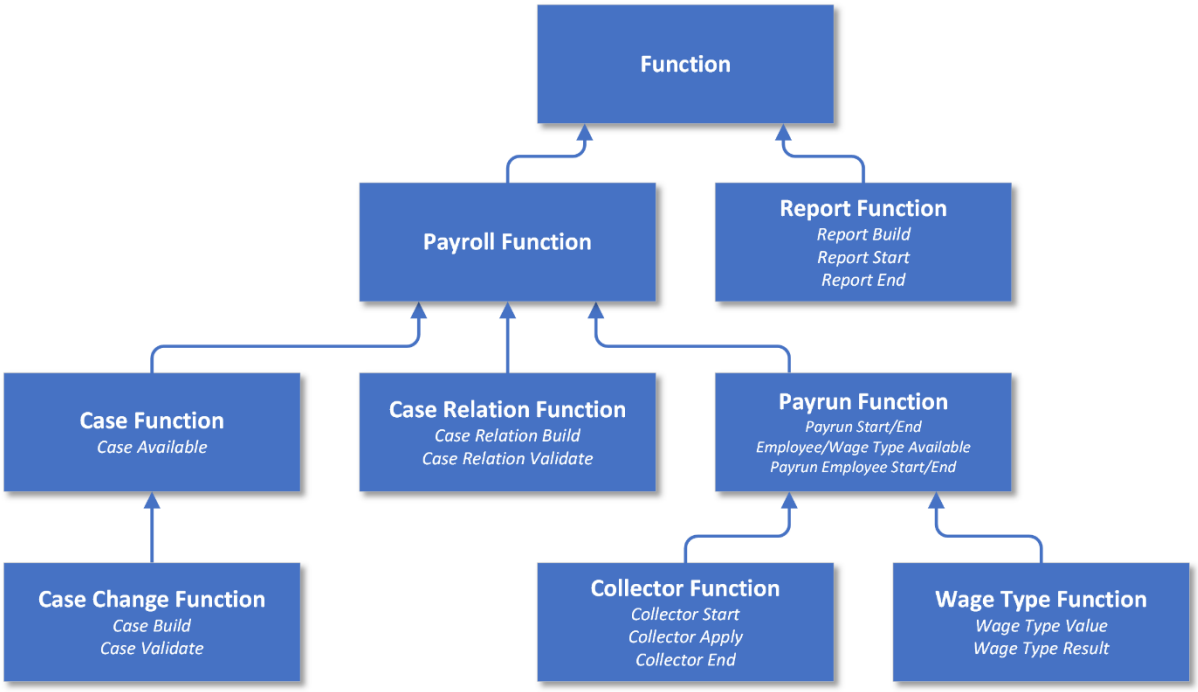


Im Onboarding werden Scripts entwickelt und mit der *Payroll Console* zur Payroll REST API übertragen. Beim Einspielen der Scripts werden diese in die Maschinensprache übersetzt (kompiliert). Zur Laufzeit werden die binären Scripts vom Payroll Prozess geladen und ausgeführt. Das Script kann währenddessen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

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

6.1 Funktionen

Eine Funktion ist eine programmierte Sequenz, welche von einem Backendprozess an spezifischen Stellen aufgerufen werden. Die Funktionen sind in folgende Bereiche gegliedert:



Die verfügbaren Scripting Funktionen sind:

Objekt	Funktion	Beschreibung	Typ	Überlagerung
Case	Available	Testen ob Case verfügbar ist	bool?	1. Top-Down
Case	Build	Case erstellen	bool?	1. Top-Down
Case	Validate	Testen ob Case gültig ist	bool?	1. Top-Down
Case Relation	Build	Related Case erstellen	bool?	1. Top-Down
Case Relation	Validate	Testen ob Related Case gültig ist	bool?	1. Top-Down
Payrun	Start	Payrun Start	bool?	-
Payrun	EmployeeAvailable	Testen ob Mitarbeiter verfügbar ist	bool?	-
Payrun	EmployeeStart	Payrun Employee Start	bool?	-
Payrun	WageTypeAvailable	Testen ob Lohnart verfügbar ist	bool?	-
Payrun	Employee End	Payrun Employee Ende	-	-
Payrun	End	Payrun Ende	-	-
Collector	Start	Collector Start	-	1. Top-Down
Collector	Apply	Lohnart Wert anwenden	decimal?	1. Top-Down
Collector	End	Collector 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	-	-

Mit der “1. Top-Down“ Überlagerung wird gewährleistet, dass von den Payroll Regulierungen die erste Funktion verwendet wird, welche einen Wert liefert. Durch Rückgabe eines undefinierten Wertes wird die Funktion der zugrundeliegenden Regulierung ausgeführt.

Mit der “* Bottom-Up“ Überlagerung werden Daten zusammengetragen, beginnend bei der untersten Regulierungsschicht und der Möglichkeit, Werte in der übergeordneten Regulierung zu überschreiben.

Im Kapitel *Payroll Script* sind beiden Szenarien veranschaulicht.

Weitere Informationen zur Scripting API finden sich im *Dokument Payroll.Backend.chm* (kompilierte HTML Hilfedatei).

6.1.1 Funktionen erweitern

Die eingebauten Funktionen lassen sich mit Scripts erweitern. Dafür bestehen zwei Möglichkeiten in C#:

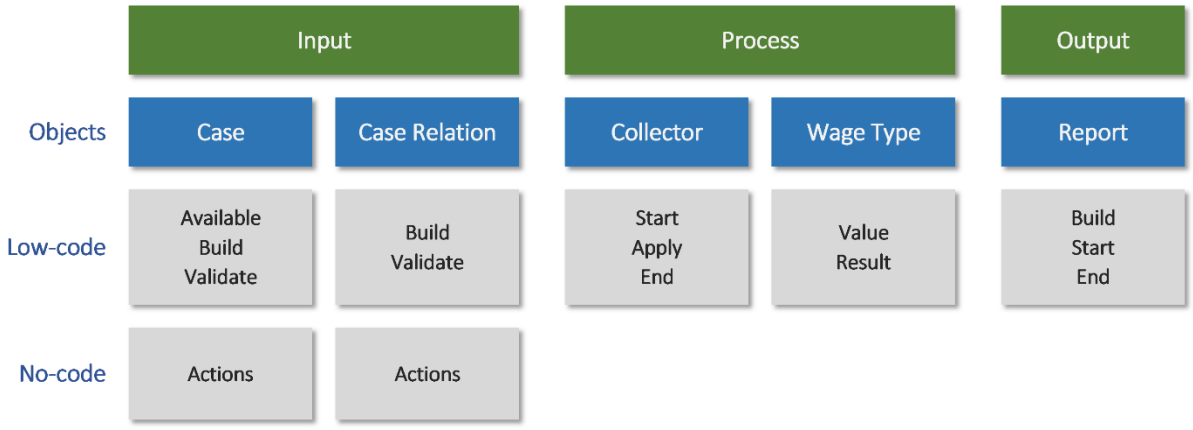
- Extension Methoden
- Partielle Klassen

Insbesondere die partiellen Klassen sind geeignet, um den Business-Code in eigenes Klassenmodell auszulagern.

Im Case-Scripting bieten Extension Methoden die Möglichkeit, die Erstellung und Validierung pro Case Field zu bestimmen.

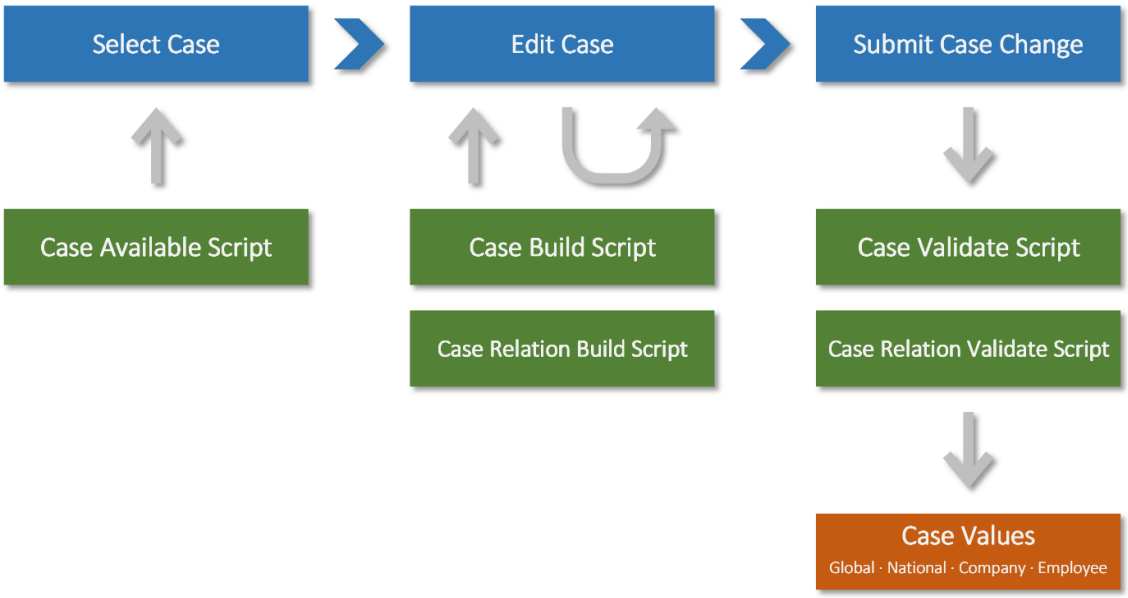
6.2 Regulation Scripting

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



6.3 Case Scripting

Der Lebenszyklus eines Lohnfalles besteht aus den Phasen Verfügbarkeit, Generierung und Übermittlung. In der Generierungsphase wird der Case initial erstellt und bei jeder Datenänderung neu aufgebaut. Es entsteht folgende Verarbeitungssequenz:

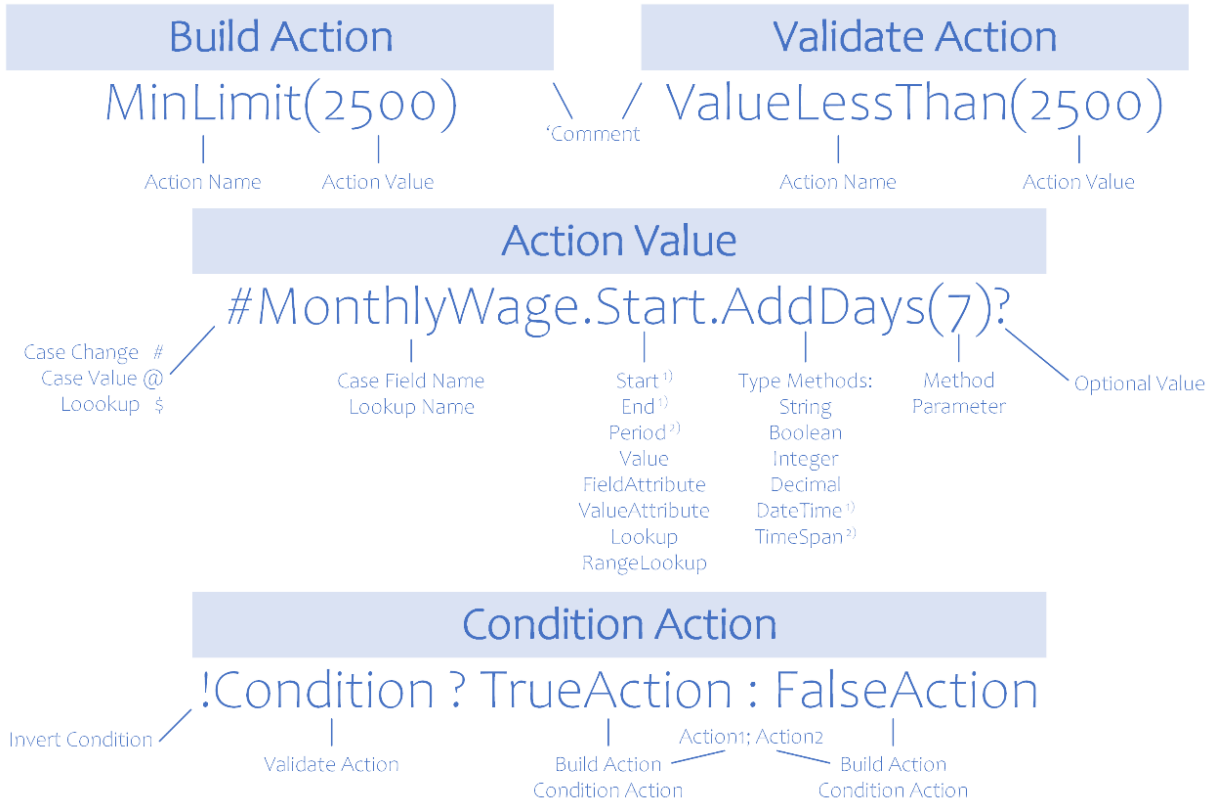


6.3.1 Case Actions

Case Actions bieten die Möglichkeit die Case Funktionen ohne Programmierkenntnisse zu steuern.
Case Actions bieten folgende Funktionen:

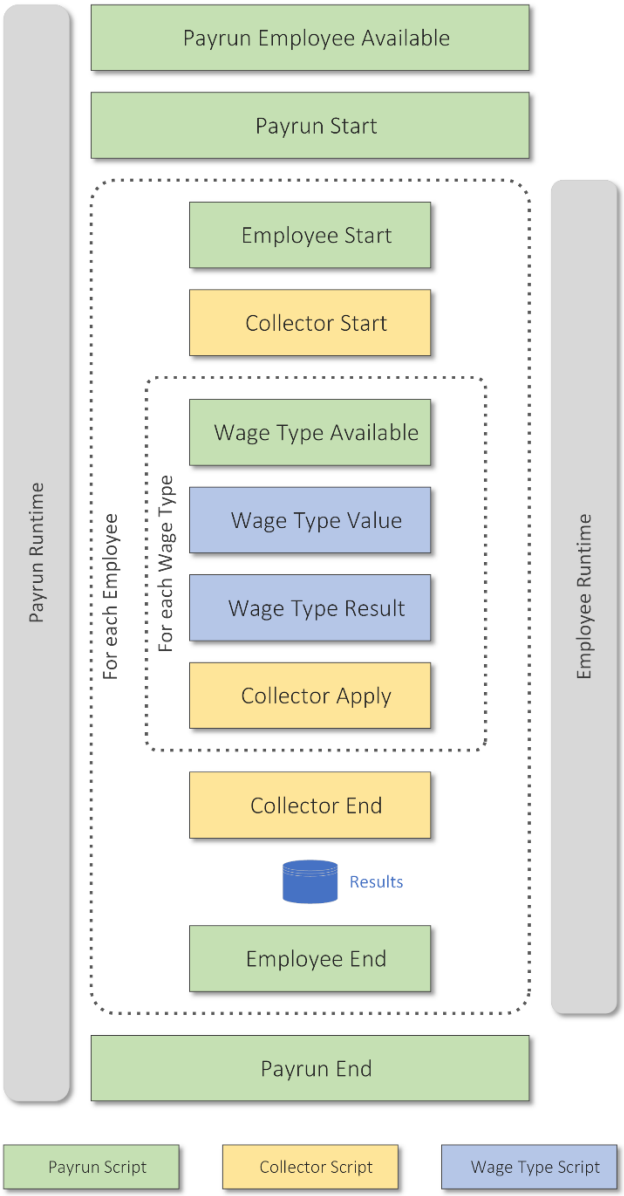
- Zugriff auf Mitarbeiterdaten und Lookups
- Werte bedingt setzen
- Werteüberprüfungen
- Steuerung der Eingabefelder

Die folgende Übersicht zeigt den Befehlssatz von Case Actions:



6.4 Payrun Scripting

Im Payrun steuern folgende Scripting Funktionen den Lohnlauf:



Zur Laufzeit dienen die *Payrun Runtime* und *Employee Runtime* als Gefäss, um Daten zwischen den Objekten *Payrun*, *Collector* und *Wage Type* auszutauschen. Als Beispiel kann ein Berechnungswert einer Lohnart durch die *Employee Runtime* von einer Folgelohnart genutzt werden.

Ist die Lohnart einem oder mehreren Kollektoren zugeordnet, wird dessen Wert (*Wage Type Value/Result*) auf die Kollektoren angewendet (*Collector Apply*). Für Sonderfälle lässt sich die Kollektorzuordnung dynamisch deaktivieren.

6.5 Payrun Laufzeitwerte

Während des Lohnlaufes können Werte zwischen den Funktionen ausgetauscht werden. Es wird zwischen Payrun- und Employee-Laufzeitwerten unterschieden. Die Payrun-laufzeitwerte bestehen über den gesamten Lohnlauf, die Employee-Laufzeitwerte bestehen jeweils pro Mitarbeiter. Am Ende des Lohnlaufes können alle Laufzeitwerte für weitere Verarbeitungen genutzt werden.

Übersicht der Payrun Laufzeitwerte:

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

6.6 Lohnlauf Logs

Scripting Funktionen können Logeinträge nach verschiedenen Stufen erstellen. Die Logs sind dem Mandanten zugeordnet und lassen sich über die API auswerten. Mit einem Startparameter des Payrun Jobs kann bestimmt werden, welche Log-Stufen zu protokolieren sind.

Anmerkung: die Logs der Payroll Engine Applikationen werden im Systemverzeichnis der Programmdaten geführt *(%ProgramData%*).

6.7 Scripts in der REST API

Scripts werden als Objektwerte geführt (z.B. Wage Type *ValueExpression* und *ResultExpression*) und werden beim Einspeisen des Objektes (POST) in Maschinencode kompiliert. Bei einem Syntaxfehler liefert der Payroll Dienst den Fehlercode “422 Unprocessable Entity“.

6.8 Geteilte Scripts

Das Regulierungsobjekt Script beinhaltet Quellcode, welcher von verschiedenen Scripting Objekten genutzt wird. Der Quellcode kann bestehende Funktionen erweitern (C# Extension Methods) oder zusätzliche Tools zur Verfügung stellen. Diese Erweiterungen werden bei der Kompilierung des Zielobjektes (z.B. WageType) integriert und sind nicht miteinander verknüpft. Aktualisierungen im Script Objekt tangieren das Zielobjekt nicht.

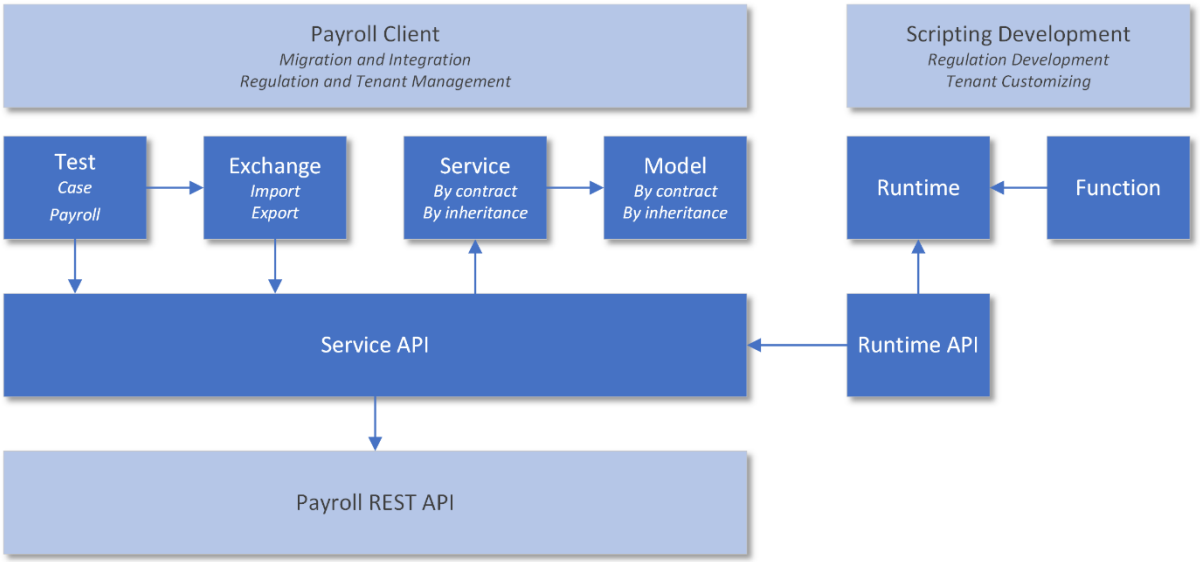
Um diese Scripts zu aktualisieren, bieten folgende Objekte einen *Rebuild* Endpunkt an:

- Case
- Case Relation
- Collector
- Wage Type
- Report
- Payrun

7 Client Services

Die Client Services sind zusätzliche Dienste die Nutzung der Payroll API effizienter zu nutzen für

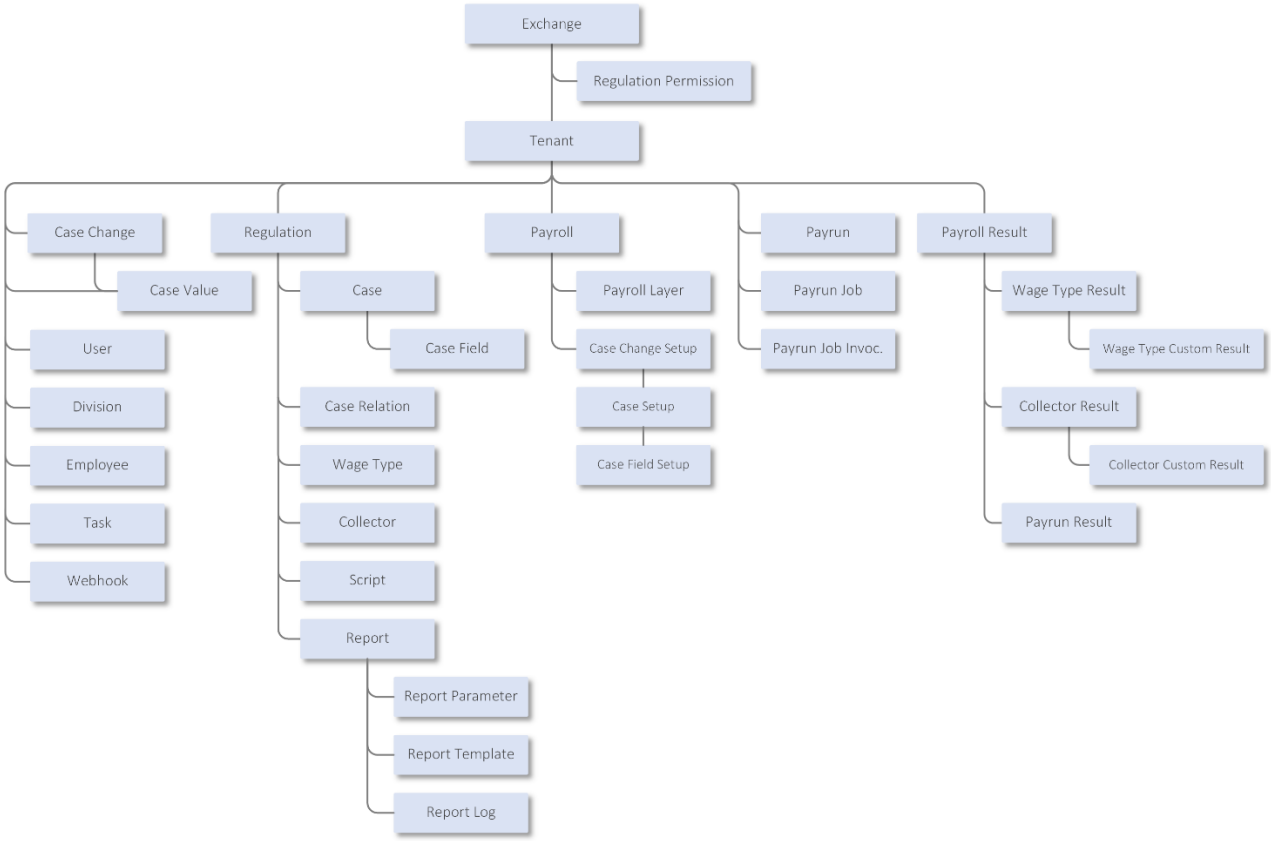
- Migrationen und Integrationen (z.B. Schnittstellen)
- die Verwaltung von Regulierungen und Mandanten
- das Testen von Regulierungen
- die Entwicklung von Scripts



Model	Model der API-Objekte (Swagger Schema) erweitert mit Exchange relevanten Werten und Eigenschaften. Ein Modelobjekt ist kopier-/vergleichbar und kann durch Vererbung oder Contract/Interface genutzt bzw. erweitert werden.
Service	Zugriff auf die API-Endpunkte (Swagger Endpoints). Ein Service kann durch Vererbung oder Contract/Interface genutzt bzw. 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 beinhaltet alle Payroll API Objekte für den Datenaustausch:

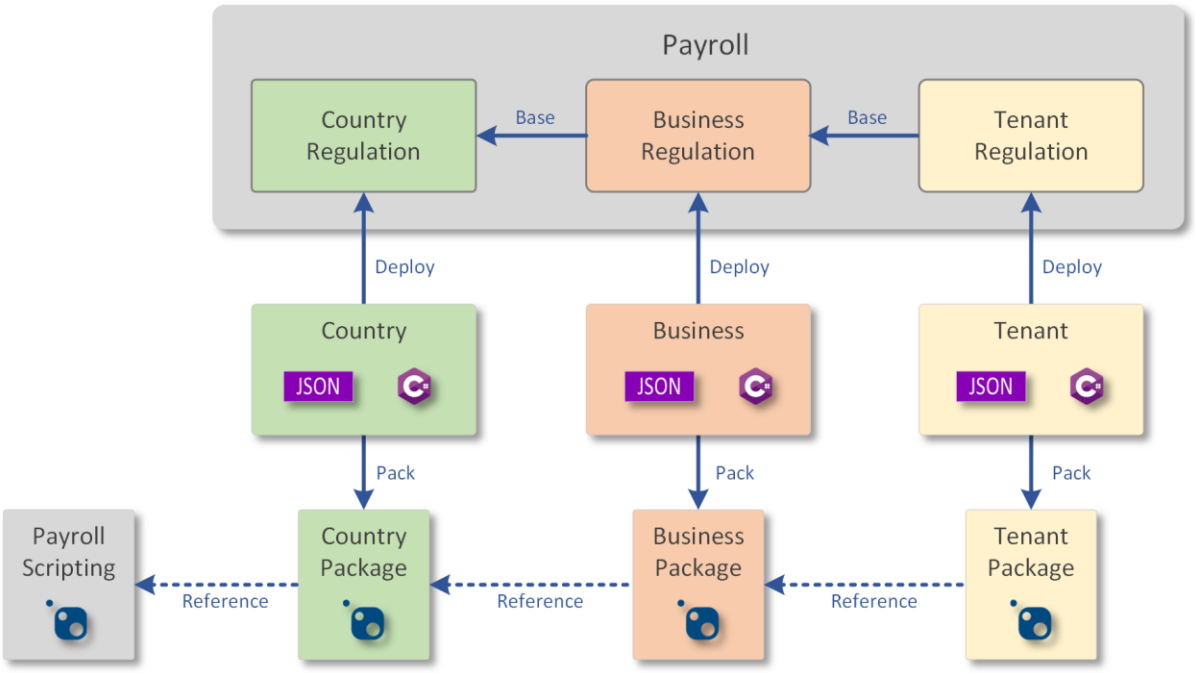


Für die Bearbeitung des Models stehen verschiedene Tools zur Verfügung (z.B. Import und Export), welche auch auf reduzierte Modellbereiche funktionieren. Zusätzlichen bestehen Klassen, welche die Modell-Verarbeitungssequenz übernehmen und dem Anwender den direkten Objektzugriff ermöglichen.

7.1.2 Regulierungen entwickeln und bereitstellen

Die Regulierung wird 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 Landes- und Branchen-Regulierungen vorausgesetzt. Neben der Dokumentation wird der NuGet-Mechanismus verwendet, um Funktionen bereitzustellen.

Die folgende Abbildung zeigt die Verwendung und Bereitstellung von Regulation Packages, sowie die Bereitstellung der Regulierungen für das Lohnlaufsystem:



7.1.3 Mandanten Namespaces

Die Import, Export und Testfunktionen bieten die Möglichkeit, Vorlagen auf verschiedene Mandanten anzuwenden. Dabei dient die Identifikation des Mandanten als Namespace und wirkt sich auf alle Objekte aus, deren Bezeichnung mit dem Namespace-Bezeichner beginnen. Dieses Verhalten unterstützt folgende Anwendungsfälle:

- Vergleichen von verschiedene Zustandsvarianten
- Zusammenführen von Mandanten
- Regulation White Labeling

Im folgenden Beispiel wird beim Mandant *MySolution* der Namespaces *MyCopy* angewendet:

- Mandant Identifikation *MySolution → MyCopy*
- Case Name *MySolution.MyCase → MyCopy.MyCase*
- Collector Name *MySolution.MyCollectior → MyCopy.MyCollectior*
- …und so weiter für alle Namen und Namenreferenzen

8 Clients

Basierend auf den Client Services bestehen folgende Clients:

- Payroll Konsole: Import, Export und Test von Payrolls
- Web Applikation: Visuelle Verwaltung der Payroll Engine

8.1 Payroll Konsole

Die Payroll Konsolenapplikation 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
- Mandant Löschen
- Payroll API-Abfrage
- Payroll Report in verschiedenen Formaten

8.2 Web App

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

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

9 Quelle

Die Payroll Engine mit allen Beispielen ist auf GitHub gehostet:

https://github.com/Payroll-Engine