Funktionsweise Ermittlung Consentstatus im gICS

Autor: Lars Geidel, Peter Penndorf, Arne Blumentritt

Datum: 01.09.2024

Allgemeines

Die Frage nach dem Consentstatus wird auf Basis der „SignedPolicies“ ermittelt. Diese sind eigentlich virtuelle Objekte, die aus Performancegründen eingeführt wurden. Die entsprechende Tabelle dazu ebenso. Mit der Cacheversion die Consente cached ist die Tabelle obsolet geworden und könnte samt des Ladens der SignedPolicies entfernt werden. Diese sind ein Mischung aus Modulstatus, zum Modul zugehörige Policies und Consentdatum. Sie können also eigentlich beim Start aus ebendiesen Objekten (die dann alle gecached sind) erstellt werden.

Bedeutung von Expiration Dates

Weitere zur Bestimmung des Consentstatus wichtige Informationen sind die diversen Ablaufdaten, diese werden im Consent in dem transienten Feld `consentDateValues` (ConsentDateValues) gespeichert und beim dynamisch beim ersten Zugriff berechnet (`Consent.calculateConsentDateValues`). Dabei wird für jede Ebene (Consent, Modul, Policy) Der als nächste „fällige“ Zeitpunkt aller Ebenen bis zu diesem Objekt genommen. Der jeweilige Zeitpunkt kann sich aus Consentdatum + Laufzeit ergeben oder als fester Zeitpunkt konfiguriert werden.

ExpirationDates/Periods können an folgenden Stellen gesetzt werden

- Ablaufdatum Domäne
- Ablaufzeitraum Domäne
- Ablaufdatum Vorlage
- Ablaufzeitraum Vorlage
- Ablaufdatum IC
- Ablaufdatum Policy eines Moduls
- Ablaufzeitraum Policy eines Moduls
- Ablaufdatum Modul einer Vorlage
- Ablaufzeitraum Modul einer Vorlage

Bedeutung SignerId / VirtualPerson

Extern gibt es zur Identifikation von Personen (zu einem Consent) beliebige IDs (Pat-ID, Fallnummer, Studien-PSN, ...), im gICS intern gibt es VirtualPersons. Jede VirtualPerson ist durch eine bestimmte feste (im Sinne von: wenn angelegt, ändert sie sich nicht mehr) Menge von IDs definiert. Jeder Consent ist mit mindestens einer VP verknüpft. Mindestens, da bei Hinzufügen einer ID zu anderen (z.B. Fallnummer zu einem Consent) eine neue VP mit dieser neuen, erweiterten Menge an IDs angelegt wird. Es sind dann beide VPs mit diesem Consent verbunden. Das ermöglicht sehr spezifische Abfragen von Consenten.

Vorgehen zur Ermittlung des Consent Status und hochgradige Konfigurierbarkeit des gICS

Abgesehen von der einmaligen Berechnung der Ablaufdaten (siehe Expiration Dates) ist die Ermittlung des Consentstatus im Prinzip trivial: alle passenden SignedPolicies werden übereinandergelegt und das Endergebnis des Stapels ist der gesuchte Consentstatus. Diese „Berechnung" erfolgt in einer von zwei Funktionen (`DAO.getConsentStatusTypeFromTo` und `DAO.getCurrentPolicyStatesForSignerId`) – je nachdem, ob man nur den Status oder die SignedPolicies haben möchte. Jede Interfacefunktion benutzt je nach Rückgabewert eine der beiden „Berechnungsfunktionen". Diese sind daher inhaltsgleich, benutzen aber intern jeweils eine weitere andere wichtige Funktion: `DAO.checkPolicies` bzw. `DAO.getCurrentPolicy`. Die beiden sind – bezüglich Ablaufsteuerung – ebenfalls inhaltsgleich.

Die Komplexität ergibt sich durch die hohe Konfigurierbarkeit so ziemlich jeden Aspektes.

Ablauf

1. Es werden alle SignedPolicies gesammelt, die mit den übergebenen SignerIds in Zusammenhang stehen (Configmöglichkeit: `CheckConsentConfig.idMatchingType`, `CheckConsentConfig.useAliases`), bis zum Requestzeitpunkt bekannt waren (Configmöglichkeit: `CheckConsentConfig.useHistoricalData`) und zu den übergebenen Policies passen (Configmöglichkeit: `CheckConsentConfig.ignoreVersionNumber`).

2. Diese werden nach Datum geordnet (Configmöglichkeit: `DomainConfig.policyConfig.takeHighestVersionInsteadOfNewest`).

3. Der Stapel wird durchlaufen, Startstatus ist `unknown`, der Ergebnisstatus ist der letzte gefundene (Configmöglichkeiten: `DomainConfig.policyConfig.permantRevoke`), dabei werden weitere Vorkommen von „unknown" ignoriert (Configmöglichkeiten: `CheckConsentConfig.unknownStateIsConsideredAsDeclined`)

Relevante Configeinträge

DomainConfig.policyConfig

permantRevoke – default `false` – wenn in dem SignedPolicstapel ein „Nein" auftaucht, ist das Gesamtergebnis auf jeden Fall „Nein"

takeHighestVersionInsteadOfNewest – default `false` – der Stapel wird nicht nach Datum sondern nach PolicyVersionsnummer sortiert

takeMostSpecificValidityInsteadOfShortest – default `false` – es soll für die Berechnung der Ablaufdaten immer das Ablaufdatum des Objektes benutzt werden, nicht das für `Consent.consentDateValues` berechnete (siehe Expiration Dates) – ist aktuell ohne Funktion / nicht umgesetzt

CheckConsentConfig:

idMatchingType – wie sollen die SignedPolicies mittels übergebener SignerIDs gefiltert werden? (siehe SignerId / VirtualPerson)

- `AT_LEAST_ONE` – default – mindestens eine Id muss in der VirtualPerson enthalten sein
- `AT_LEAST_ALL` – alle Ids müssen in der VP enthalten sein
- `EXACT` – alle Ids müssen in der VP enthalten sein und keine weiteren

ignoreVersionNumber – default `false` – übergebene Versionsnummern der Policy ignorieren (also nur den Policynamen benutzen)

unknownStateIsConsideredAsDeclined – default `false` – ConsentStatus.unknown wird als ConsentStatus.declined behandelt

requestDate – default `now` – Anfragezeitpunkt (für historische Anfragen)

useAliases – default `false` – sollen bei der Bestimmung der VirtualPersons Aliase zu den üebergebenen SignerIds berücksichtigt werden? Wird aktuell bei IdMatchingType.EXACT nicht beachtet!

useHistoricalData – default `false` – soll bei historischen Anfragen der Stand von damals oder der aktuelle Stand benutzt werden?

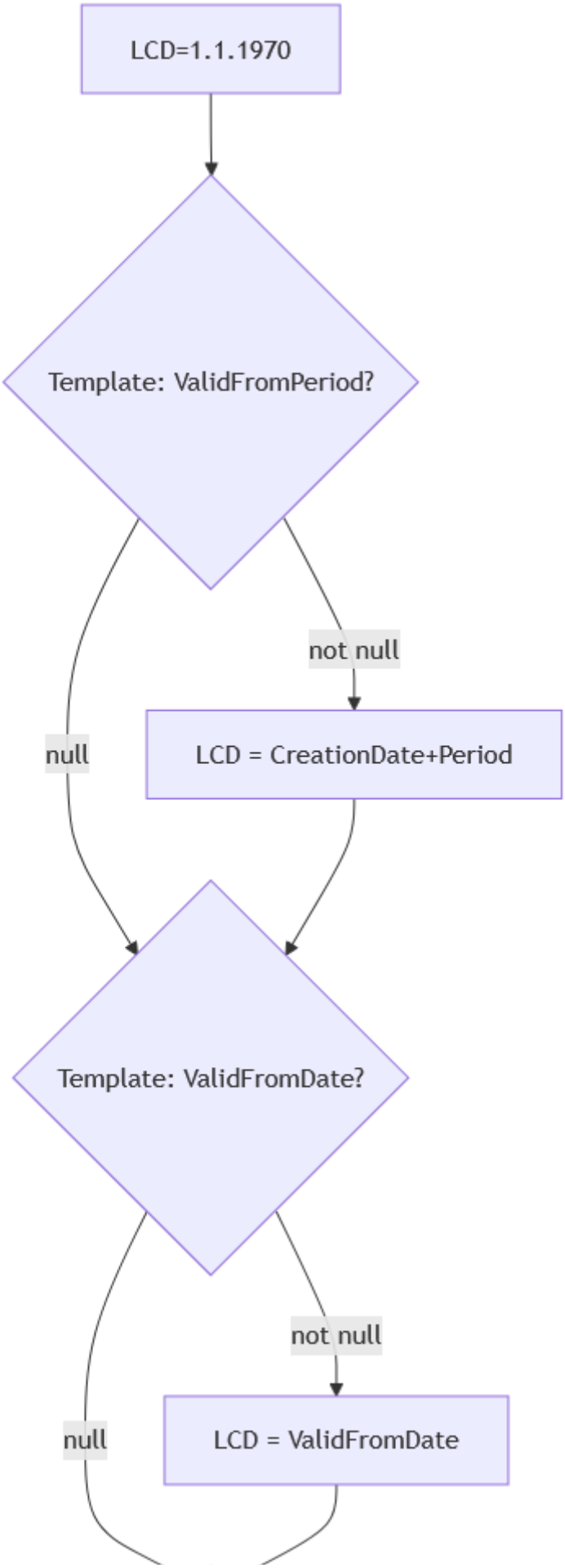
Berechnung des LegalConsentDates

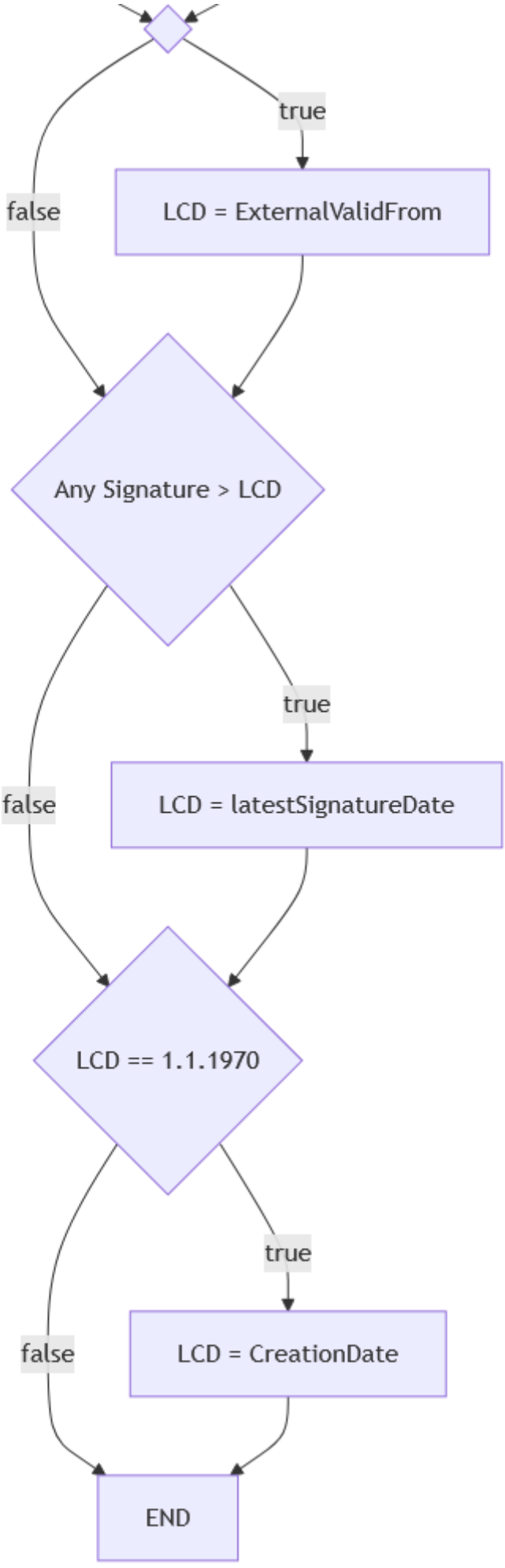
Das Legal-Consent-Datum sagt aus, ab wann ein IC als gültig betrachtet wird und damit in die Abfragelogik einbezogen wird. Das Datum berechnet sich aus vielen verschiedenen Enflussfaktoren. Wie genau wird im Folgenden beschrieben.

Einflussfaktoren

- **Anlegedatum:** das Datum an dem ein IC angelegt/erstellt wurde
- **Unterschriften-Datumsangaben** alle Datumsangaben von Unterschriften
- **externes Gültigkeitsdatum (ValidFrom)** ein von extern vorgegebenes Gültigkeitsdatum
- **Gültigkeitsdatum (aus Template)** das feste Gültigkeitsdatum aus den ValidFromProperties des Templates
- **Gültigkeitsfrist (aus Template)** Angabe einer Zeitspanne, ab Anlegen eines ICs, wie Lange ein IC noch nicht gültig sein soll. Kommt aus den ValidFromProperties des Templates

Ablaufdiagramm





LCD = LegalConsentDate { width=75% }

Credits

Concept and implementation: L. Geidel
Web-Client: A. Blumentritt, M. Bialke, F.M.Moser
Docker: R. Schuldt
TTP-FHIR Gateway für gICS: M. Bialke, P. Penndorf, L. Geidel, S. Lang, F.M. Moser

License

Contact: https://www.ths-greifswald.de/kontakt/

Publications

- https://doi.org/10.1186/s12911-022-02081-4
- https://rdcu.be/b5Yck
- https://rdcu.be/6LJd
- https://dx.doi.org/10.3414/ME14-01-0133
- https://dx.doi.org/10.1186/s12967-015-0545-6