

Natural for z/OS

Natural API Server

Version 9.2.5

Oktober 2026

This document applies to Natural for z/OS Version 9.2.5 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1979-2026 Software GmbH, Darmstadt, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software GmbH product names are either trademarks or registered trademarks of Software GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software GmbH and/or its subsidiaries is located at <https://softwareag.com/licenses>.

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software GmbH Products / Copyright and Trademark Notices of Software GmbH Products". These documents are part of the product documentation, located at <https://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use of this software is subject to adherence to Software GmbH's licensing conditions and terms. These terms are part of the product documentation, located at <https://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Document ID: NATMF-NATNPI-925-20261013DE

Inhaltsverzeichnis

Vorwort	v
1 Über diese Dokumentation	1
Dokumentationskonventionen	2
Online-Informationen und Support	2
Datenschutz	3
I Übersicht	5
2 Übersicht	7
Was ist der Natural API Server	8
Architektur	8
Über den Natural API Server	9
II Natural Subprogram Server auf z/OS	11
3 Einführung	13
Konzept des Natural Subprogram Server	14
Front-end Stub NATRNPI	15
Frontend	16
Gateway-Modul	16
Server-Monitor	16
Produkt-Interaktion	17
4 Installation	19
Voraussetzungen	20
Inhalt des Installationsdatenträgers für den Natural Subprogram Server	20
Installation des Natural Subprogram Server auf z/OS	21
5 Konfiguration	25
Konfigurationserfordernisse	26
Konfiguration der externen Security	27
Konfigurationsdatei	27
Konfigurationsparameter	28
Beispiel für eine Konfigurationsdatei	34
Natural Subprogram Server Datasets	34
6 Operation	35
Natural Subprogram Server starten	36
Natural Subprogram Server beenden	36
Natural Subprogram Server überwachen	37
Runtime Trace Facility	38
7 Sicherheit	41
Natural Security	42
Externe Security	42
III Natural OpenAPI Server auf Tomcat	43
8 Installation	45
Installation des Natural OpenAPI Server	46
9 Konfiguration	47
Allgemeine Konfiguration	48

Mapping von Natural-Fehlern auf HTTP-Statuscodes	49
10 Betrieb	51
Bereitstellung der Webanwendung mit Tomcat	52
Bereitstellung von OpenAPI-Ressourcen	52
11 Überwachung	55
12 Sicherheit	57
Sichere Kommunikation zwischen dem Natural OpenAPI Server und dem Subprogram Server	58
Sichere Kommunikation zwischen Client und Natural OpenAPI Server	59
Authentifizierung	60
IV Natural OpenAPI Server auf Liberty	63
13 Installation	65
Installation des Natural OpenAPI Server	66
14 Konfiguration	67
Allgemeine Konfiguration	68
Mapping von Natural-Fehlern als HTTP-Statuscodes	69
15 Betrieb	71
Bereitstellung der Webanwendung mit Liberty	72
Bereitstellung von OpenAPI-Ressourcen	73
16 Überwachung	75
17 Sicherheit	77
Sichere Kommunikation zwischen dem Natural OpenAPI Server und dem Subprogram Server	78
Sichere Kommunikation zwischen Client und Natural OpenAPI Server	79
Authentifizierung	80
V Übersicht für Entwickler	83
18 Installation des NaturalONE OpenAPI Plugin	85
19 Konfiguration des NaturalONE OpenAPI Plugin	87
20 Dienste generieren und bereitstellen	91
OpenAPI-Ressourcen generieren	92
OpenAPI-Ressourcen im Editor anzeigen	94
21 Besondere Hinweise	97
Entwicklung von Natural-Subprogrammen für Natsubsv	98

Vorwort

Diese Dokumentation beschreibt den Natural API Server für Natural for z/OS.

Übersicht
Natural Subprogram Server auf z/OS
Natural OpenAPI Server auf Tomcat
Natural OpenAPI Server auf Liberty
Übersicht für Entwickler

1 Über diese Dokumentation

■ Dokumentationskonventionen	2
■ Online-Informationen und Support	2
■ Datenschutz	3

Dokumentationskonventionen

Konvention	Beschreibung
Fettschrift	>Kennzeichnet Elemente auf einem Bildschirm.
Nichtproportionale Schrift	Kennzeichnet Namen und Orte von Diensten im Format <i>Ordner.Unterordner.Dienst</i> , Programmierschnittstellen (APIs), Namen von Klassen, Methoden und Properties in Java.
<i>Kursivschrift</i>	Kennzeichnet: Variablen, für die Sie situations- oder umgebungsspezifische Werte angeben müssen. Neue Begriffe, wenn sie erstmals im Text auftreten. Verweise auf andere Dokumentationsquellen.
Nichtproportionale Schrift	Kennzeichnet: Text, den Sie eingeben müssen. Meldungen, die vom System angezeigt werden. Programmcode.
{ }	Zeigt eine Reihe von Auswahlmöglichkeiten an, von denen Sie eine auswählen müssen. Geben Sie nur die innerhalb der geschweiften Klammern vorhandenen Informationen ein. Geben Sie nicht die Klammersymbole { } ein.
	Trennt zwei sich gegenseitig ausschließende Auswahlmöglichkeiten in einer Syntaxzeile voneinander ab. Geben Sie eine der Auswahlmöglichkeiten ein. Geben Sie nicht das Symbol ein.
[]	Zeigt eine oder mehrere Optionen an. Geben Sie nur die innerhalb der eckigen Klammern vorhandenen Informationen ein. Geben Sie nicht die Klammersymbole [] ein.
...	Zeigt an, dass Sie mehrere Auswahlmöglichkeiten desselben Typs eingeben können. Geben Sie nur die Informationen ein. Geben Sie nicht die drei Auslassungspunkte (...) ein.

Online-Informationen und Support

Produktdokumentation

Sie finden die Produktdokumentation auf unserer Dokumentationswebsite unter <https://documentation.softwareag.com>.

Zusätzlich können Sie auch über <https://www.softwareag.cloud> auf die Dokumentation für die Cloud-Produkte zugreifen. Navigieren Sie zum gewünschten Produkt und gehen Sie dann, je nach Produkt, zu „Developer Center“, „User Center“ oder „Documentation“.

Produktschulungen

Sie finden hilfreiches Produktschulungsmaterial auf unserem Lernportal unter <https://knowledge.softwareag.com>.

Tech Community

Auf der Website unserer Tech Community unter <https://techcommunity.softwareag.com> können Sie mit Experten der Software AG zusammenarbeiten. Von hier aus können Sie zum Beispiel:

- Unsere umfangreiche Wissensdatenbank durchsuchen.
- In unseren Diskussionsforen Fragen stellen und Antworten finden.
- Die neuesten Nachrichten und Ankündigungen der Software AG lesen.
- Unsere Communities erkunden.
- Unsere öffentlichen Repositories auf GitHub and Docker unter <https://github.com/softwareag> und <https://hub.docker.com/publishers/softwareag> besuchen und weitere Ressourcen der Software AG entdecken.

Produktsupport

Support für die Produkte der Software AG steht lizenzierten Kunden über unser Empower-Portal unter <https://empower.softwareag.com> zur Verfügung. Für viele Dienstleistungen auf diesem Portal benötigen Sie ein Konto. Wenn Sie noch keines haben, dann können Sie es unter <https://empower.softwareag.com/register> beantragen. Sobald Sie ein Konto haben, können Sie zum Beispiel:

- Produkte, Aktualisierungen und Programmkorrekturen herunterladen.
- Das Knowledge Center nach technischen Informationen und Tipps durchsuchen.
- Frühwarnungen und kritische Alarmer abonnieren.
- Supportfälle öffnen und aktualisieren.
- Anfragen für neue Produktmerkmale einreichen.

Datenschutz

Die Produkte der Software AG stellen Funktionen zur Verarbeitung von personenbezogenen Daten gemäß der Datenschutz-Grundverordnung (DSGVO) der Europäischen Union zur Verfügung. Gegebenenfalls sind in der betreffenden Systemverwaltungsdokumentation entsprechende Schritte dokumentiert.

I Übersicht

2 Übersicht

■ Was ist der Natural API Server	8
■ Architektur	8
■ Über den Natural API Server	9

Dieses Dokument behandelt die folgenden Themen:

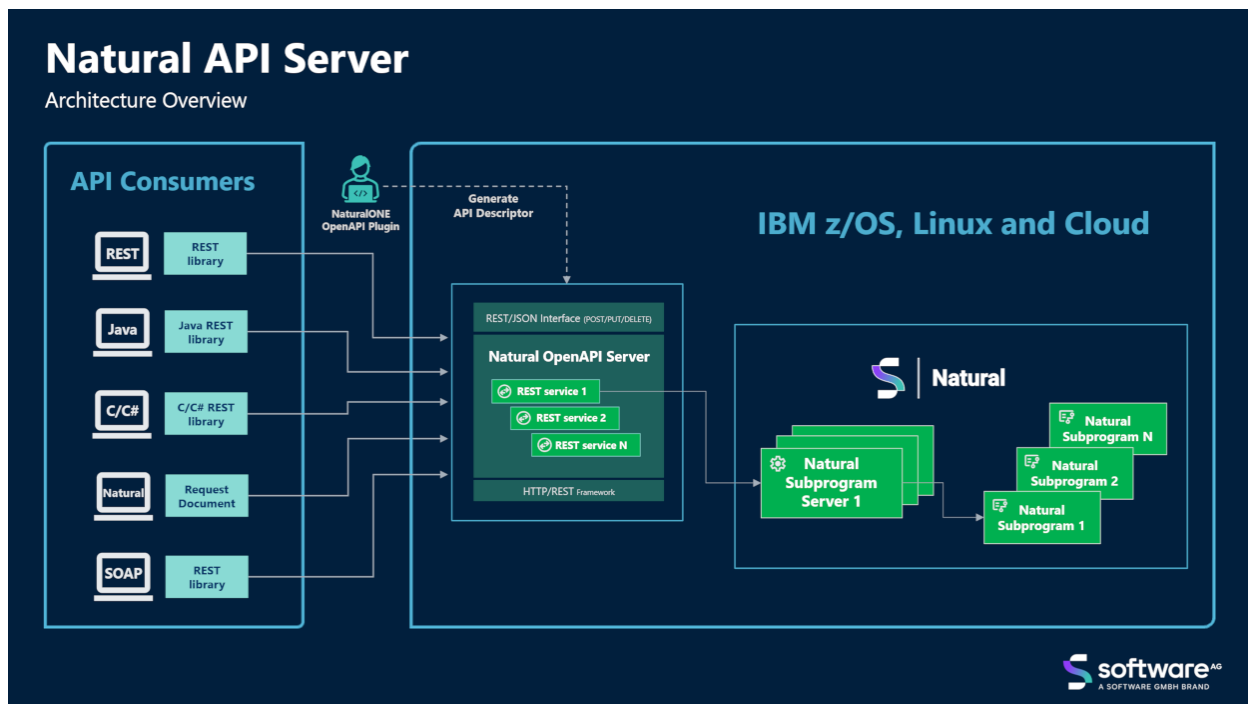
Was ist der Natural API Server

Der Natural API Server ermöglicht den Zugriff auf Natural-Subprogramme über REST API-Aufrufe, die auf dem OpenAPI-Standard basieren. Dadurch können externe Anwendungen auf eine Natural-Business Logic zugreifen und diese wiederverwenden, ohne dass bestehende Natural-Subprogramme umgeschrieben werden müssen. Die OpenAPI-Beschreibungen können automatisch generiert werden.

Architektur

Der Natural API Server besteht aus den folgenden Komponenten:

- Natural Subprogram Server for z/OS oder Linux und Cloud.
- Natural OpenAPI Server als Schnittstelle zwischen REST API Clients und dem Natural Subprogram Server. Diese Komponente läuft eigenständig als Embedded Tomcat® oder wird auf einem Java-Anwendungsserver, z.B. Apache Tomcat® oder Liberty®, bereitgestellt.
- NaturalONE OpenAPI Plugin zum Erstellen und Bereitstellen von Natural OpenAPI-Diensten.



Über den Natural API Server

Der Natural API Server ist als installierte Komponente Ihrer Natural for z/OS- oder Linux-Umgebung sowie als Eclipse-Plugin für NaturalONE verfügbar.

II Natural Subprogram Server auf z/OS

Der Natural Subprogram Server stellt Natural-Subprogramme als Dienste zur Integration in moderne Anwendungsarchitekturen bereit.



Anmerkung: Der Natural Subprogram Server stellt keine direkte REST API für Client-Anwendungen zur Verfügung. Der Zugriff erfolgt stets über den Natural OpenAPI Server, eine Middleware-Komponente, die eine REST API bereitstellt und Anfragen mithilfe eines internen Protokolls an den Natural Subprogram Server weiterleitet.

Dieses Dokument behandelt die folgenden Themen:

Einführung
Installation
Konfiguration
Betrieb
Security

3 Einführung

■ Konzept des Natural Subprogram Server	14
■ Front-end Stub NATRNPI	15
■ Frontend	16
■ Gateway-Modul	16
■ Server-Monitor	16
■ Produkt-Interaktion	17

Dieses Dokument behandelt die folgenden Themen:

Konzept des Natural Subprogram Server

Ein Natural Subprogram Server ist eine Mehrbenutzer- und Multitasking-Anwendung. Er kann Natural-Sitzungen für mehrere Benutzer hosten und deren Anfragen parallel ausführen.

Das Konzept basiert auf dem „serverisierten“ Natural-Laufzeitsystem. Seine Architektur umfasst einen Server-Frontend-Stub (Subprogram Server Stub `NATRNPI`), der das Natural-Frontend nutzt, um Natural-Sitzungen zuzuweisen und Subprogramme innerhalb dieser Sitzungen auszuführen.

Die Architektur besteht im Wesentlichen aus:

■ Frontend-Stub

Der Stub `NATRNPI` wird gestartet, um einen Natural Subprogram Server zu initialisieren. Er überwacht eingehende Anfragen und initialisiert eine Natural-Sitzung zur Ausführung des Subprogramms.

■ Frontend

Das Frontend wird (zusammen mit dem Natural-Laufzeitsystem) vom Frontend-Stub zur Initialisierung bzw. Beendigung der Sitzung sowie zur Ausführung von Anfragen aufgerufen.

■ Gateway-Modul

Das Modul `NATGWNPI` sorgt für die Interaktion zwischen dem Natural-Laufzeitsystem und dem Frontend-Stub. `NATGWNPI` ist bereits im Natural-Nukleus enthalten und wird vom Natural-Laufzeitsystem aufgerufen, um die erforderlichen Anfragedaten auszutauschen.

■ Natural-Treiber

Natural wird vom Natural-Treiber `NATOS` bzw. `NATOSL` gesteuert.



Anmerkung: `NATOSL` ist eine Erweiterung von `NATOS`, die IBMs Language Environment (LE) unterstützt. LE ermöglicht die Verwendung verschiedener Programmiersprachen innerhalb einer einzigen Anwendung. Da der Server in C implementiert ist und LE benötigt, muss das Natural-Frontend stets sowohl `NATOS` als auch `NATOSL` enthalten.

■ Server Monitor

Über einen Monitor-Task kann der Administrator die Serveraktivitäten überwachen und den gesamten Server herunterfahren usw.

Front-end Stub NATRNPI

NATRNPI ist ein Multi-User- und Multitasking-Frontend-Stub, der zur Initialisierung eines Natural-Subprogram Servers gestartet wird.

Dieses Dokument behandelt die folgenden Themen:

- [Stub-Beschreibung](#)
- [Verwendete Natural-Systemvariablen](#)
- [Natural-Ein-/Ausgabeverarbeitung](#)

Stub-Beschreibung

Die Task, die die Serverinitialisierung durchführt, `TMain`, ist der Haupt-Listener, der auf eingehende Anfragen vom Natural OpenAPI Server wartet. Sie verwaltet ein Sitzungsverzeichnis, um mehrere Clients (Benutzer) und die dazugehörigen Remote-Natural-Sitzungen zu verwalten. `TMain` hat die Aufgabe, alle eingehenden Anfragen anzunehmen und an andere Subtasks (`TWork`) weiterzuleiten. Der Ablauf ist wie folgt:

- Zunächst verbindet sich der Natural OpenAPI Server mit `TMain`, um eine Verbindung herzustellen.
- Anschließend fügt `TMain` den Client in sein Sitzungsverzeichnis ein, fügt eine neue `TWork`-Subtask hinzu und übergibt die Verbindung an `TWork`.
- `TWork` initialisiert eine neue Natural-Sitzung, verarbeitet die Anfrage, sendet das Ergebnis an den Client zurück, beendet die Natural-Sitzung und beendet sich selbst.

Jede Anfrage ist zustandslos. Für jede Anfrage wird eine separate Natural-Sitzung gestartet. Die Initialisierung der Sitzung wird jedoch beschleunigt, indem neue Sitzungen aus einer Master-Sitzung geklont werden. Bei der Serverinitialisierung wird eine Master-Sitzung initialisiert und in einem benutzerunabhängigen Zustand gehalten. Für eine neue Sitzung wird die Master-Sitzung einfach kopiert und mit der Benutzerkennung der Anfrage verknüpft.

Worker Pool

Es ist aufwendig, für jede Anfrage eine neue Worker-Task zu erstellen. Es ist möglich, erstellte Worker in einem Pool vorzuhalten, um sie für nachfolgende Anfragen wiederzuverwenden. Wenn die Konfigurationsvariable `WORKER_DELAY` auf einen Wert größer als Null gesetzt ist, wartet ein Worker nach der Ausführung einer Anfrage für die festgelegte Zeit in Sekunden. Der Worker-Pool passt seine Größe automatisch an die Auslastung des Servers an. Bei vielen parallelen Anfragen vergrößert sich der Pool, um die erhöhte Last zu bewältigen. Wenn die Auslastung sinkt, verkleinert sich der Pool entsprechend.

Verwendete Natural-Systemvariablen

Innerhalb einer Natural OpenAPI Server-Sitzung werden die folgenden Natural-Systemvariablen verwendet:

- *TPSYS enthält den Wert SERVSTUB.
- *DEVICE enthält den Wert BATCH.
- *SERVER-TYPE enthält den Wert CALLNAT.

Natural-Ein-/Ausgabeverarbeitung

Das aufgerufene Subprogramm darf keine Bildschirm-Ein-/Ausgaben durchführen. Falls dennoch eine Bildschirm-Ein-/Ausgabeverarbeitung stattfindet, schreibt der Server den Bildschirminhalt in seine Trace-Datei und bricht die Bildschirm-Ein-/Ausgabeverarbeitung ab.

Frontend

Das für einen Natural-NPI-Server erforderliche Natural-Frontend ist ein Natural-Batch-Frontend-Treiber, der Language Environment-fähig sein sollte. Weitere Informationen finden Sie in den Beispiel-Installationsjobs.

Gateway-Modul

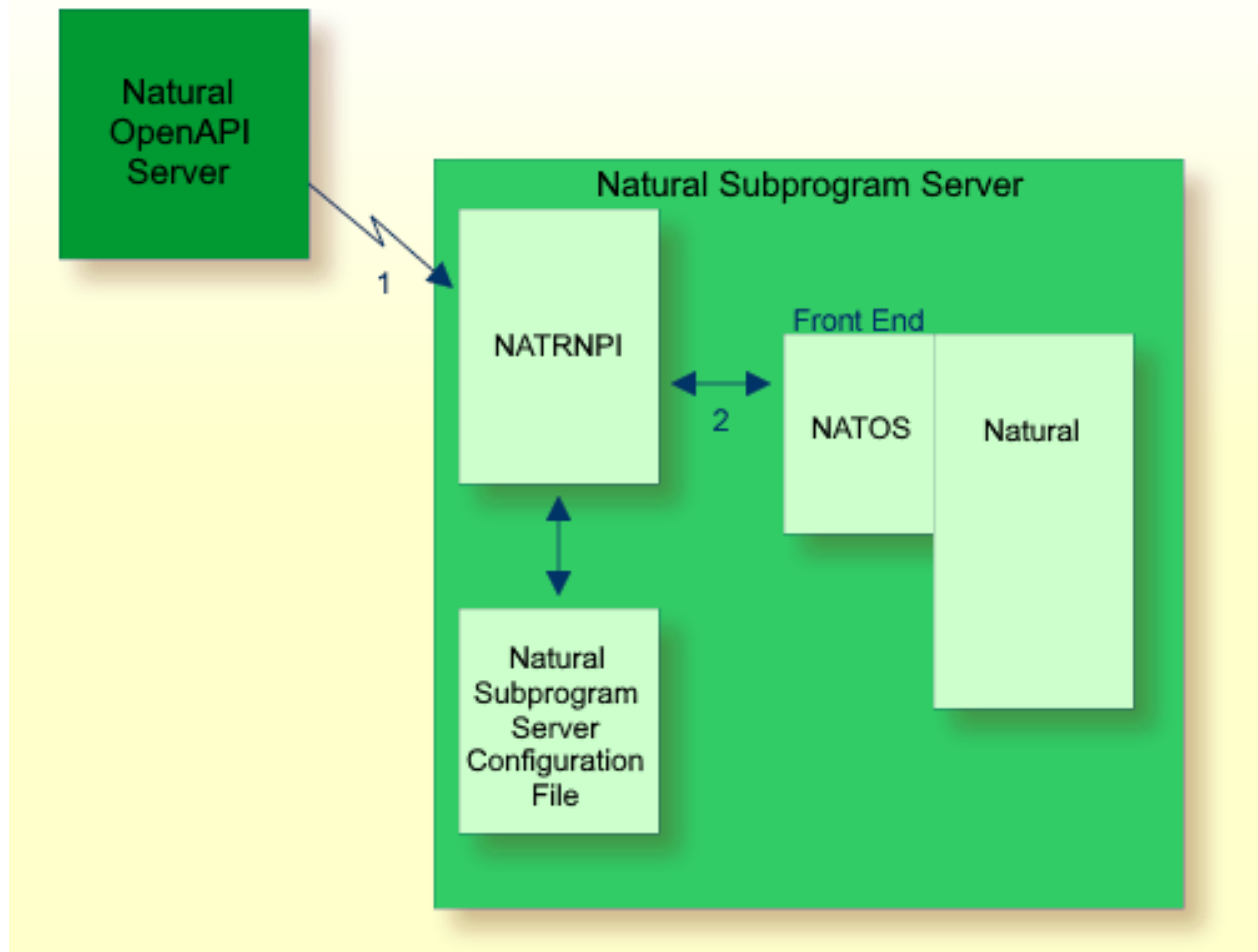
Das Gateway-Modul NATGWNPI ist bereits im Natural-Nukleus enthalten.

Server-Monitor

Damit der Administrator den Status des Natural NPI Servers überwachen kann, steht eine Monitor-Task zur Verfügung, die beim Serverstart automatisch initialisiert wird. Mithilfe der Monitor-Kommandos kann der Administrator die Serveraktivitäten steuern, den Server vollständig beenden usw. Weitere Informationen finden Sie unter *Natural API Server > Natural Subprogram Server auf z/OS > Betrieb > Überwachung des Natural API-Servers > [Monitor-Kommunikation](#)*.

Produkt-Interaktion

Die folgende Grafik veranschaulicht die Interaktion zwischen dem Natural OpenAPI Server und einem Natural NPI Server.



1. Der Natural OpenAPI Server (der Client) empfängt einen REST-Aufruf. Er wandelt diesen in ein proprietäres Protokoll um und sendet ihn über die beim Natural OpenAPI Server angegebene Portnummer an den Natural OpenAPI Server.
2. Der Natural Subprogram Server initialisiert (klont) eine Natural-Sitzung unter Verwendung des Natural-Frontend, das Sie mit dem Konfigurationsparameter `FRONTEND_NAME` des Natural NPI Servers angegeben haben. In diesem Beispiel ist dies NATOS.

4 Installation

■ Voraussetzungen	20
■ Inhalt des Installationsdatenträgers für den Natural Subprogram Server	20
■ Installation des Natural Subprogram Server auf z/OS	21

In diesem Dokument wird beschrieben, wie der Natural Subprogram Server auf z/OS installiert wird.

Dieses Dokument behandelt die folgenden Themen:

Voraussetzungen

- Die derzeit gültige Version von Natural für z/OS muss installiert sein. Weitere Informationen finden Sie unter <https://empower.softwareag.com/>.



Anmerkung: Alle anwenderseitig erstellten Exits, die nicht in Natural geschrieben wurden und in einer Subprogram Server-Umgebung verwendet werden, müssen eintrittsinvariant (reentrant) und threadsicher sein (d. h. in einer Multitasking-Umgebung lauffähig).

- Die Natural Subprogram Server Version 1.1.1 ist kompatibel mit Natural for z/OS Version 9.2.5 oder höher.
- Um die Entstehung von Endlosschleifen in Benutzerprogrammen zu verhindern, die unter dem Natural Subprogram Server ausgeführt werden, müssen Sie einen angemessenen Wert für den Natural-Profilparameter MT (Maximale CPU-Zeit) angeben.

Inhalt des Installationsdatenträgers für den Natural Subprogram Server

Der Installationsdatenträger enthält die in der folgenden Liste aufgeführten Datasets. Die Reihenfolge der Datasets und die Anzahl der benötigten Bibliotheksblöcke sind im *Product Delivery Report*, der dem Installationsdatenträger beiliegt.

Dataset-Name	Inhalt
NPIvrs.OBJS	Enthält die Objektmodule des Natural Subprogram Server.
NPIvrs.JOBS	Beispiel für Installationsjobs.

In diesem Dokument steht die Notation `vrs` in dem Dataset-Namen für die jeweilige Produktversion (siehe auch *Version* im *Glossar*).

Die aktuell gültigen Versionen finden Sie unter <https://empower.softwareag.com/>.

Installation des Natural Subprogram Server auf z/OS

Die Installation des Natural Subprogram Server auf z/OS wird in den folgenden Schritten beschrieben:

Schritt 1: Natural-NPI-Server-Konfigurationsdatei erstellen

(Job I009, Step 9110)

Schritt 9110 erstellt ein NPICONFG-Beispiel für den Natural Subprogram Server.

Die folgenden Parameter der Konfigurationsdatei müssen definiert werden. Bei den übrigen Parametern können die Standardwerte verwendet werden:

FRONTEND_NAME	Gibt den Namen des in Schritt 3 generierten Natural Subprogram Server-Frontend-Moduls an.
PORT_NUMBER	Gibt die TCP/IP-Portnummer an, über die eine Verbindung zum Server hergestellt werden kann.

Weitere Informationen zu den Parametern finden Sie unter *Natural API Server > Natural Subprogram Server auf z/OS > Konfiguration > [Konfigurationsparameter](#)*.

Schritt 2: Objektmodule in die Natural Subprogram Server-Ladebibliothek verlinken

(Job I058, Step 9120)

Die Natural Subprogram Server-Objektmodule müssen zusammen mit den erforderlichen Laufzeiterweiterungen Ihrer Batch-Installationen zu ausführbaren Lademodulen verlinkt werden.

Siehe Beispieljob NPI058 im Dataset NPIvrs.JOBS.

Schritt 3: Frontend-Modul für den Natural NPI Server erstellen

(Job I060, Steps 9110, 9120, 9130)

- Job I060, Step 9110: Starten Sie das Batch-Programm IEBUPDATE und speichern Sie das Parametermodul NPIPARM.
- Job I060, Step 9120: Assemblieren und verlinken Sie das Parametermodul NPIPARM.
- Job I060, Step 9130: Verlinken Sie das Natural-Frontend-Modul.



Anmerkungen:

1. Wenn Sie Natural-Add-on-Produkte für den Zugriff auf IBM-Datenbanken installiert haben, z. B. Natural for dB2, Natural for DL/I oder Natural for VSAM, müssen Sie den Link-Job so anpassen, dass ATRRCSS aus SYS1.CSSLIB eingebunden wird, um die Verwendung der Funktionen SYNCPOINT bzw. ROLLBACK zu ermöglichen.
2. Das Natural-Frontend muss für eintrittsinvariante (reentrant) oder wiederholte Ausführung verlinkt werden.

Schritt 4: JCL für den Serverstart erstellen

(Job I200, Step 9115)

Die Beschreibung befindet sich im Abschnitt [Konfiguration](#). Siehe dort das Beispiel-Member NPISTART im Dataset NPIvrs.JOBS.

Beispiel:

```
//          PROC  SRV=SAGNPI
//NDV       EXEC  PGM=NATRNPI,
// REGION=4000K,TIME=1440,PARM='POSIX(ON),TRAP(ON,NOSPIE)/&SRV'
//STEPLIB   DD    DISP=SHR,DSN=NPIvrs.LOAD
//          DD    DISP=SHR,DSN=SAGLIB.SMALOAD
//CEEDUMP   DD    SYSOUT=X
//CMPRINT   DD    SYSOUT=X
//STGCONFIG DD    DISP=SHR,
//          DSN=NPI.CONFIG(&SRV)
//STGTRACE  DD    SYSOUT=X
//STGSTDO   DD    SYSOUT=X
//STGSTDE   DD    SYSOUT=X
```



Anmerkung: Das Natural Subprogram Server-Konto muss in den z/OS Linux System Services (OE-Segment) definiert sein. Ist das Server-Konto nicht definiert, beendet der Server den Vorgang mit den Systemmeldungen U4093 und CEE5101C in der Trace-Datei.

Schritt 5: Die NPI Clients müssen in Natural Security definiert werden

Falls Natural Security (NSC) installiert ist:

- Die zu Beginn gültige Benutzerkennung (ID) des Natural Subprogram Server (Standardkennung ist SYSTEM) muss in Natural Security mit einer gültigen Standardbibliothek definiert werden. Weitere Informationen siehe *Natural API Server > Natural Subprogram Server auf z/OS > Konfiguration > Konfigurationsparameter > [INITIAL_USERID](#)*.
- Jede Client-Benutzerkennung muss in Natural Security definiert werden.

Wenn die zu Beginn gültige Benutzerkennung nicht definiert ist, bricht die Initialisierung des Natural Subprogram Server mit der Fehlermeldung NAT0856 ab.

Wenn ein NPI-Client nicht definiert ist, gibt der Server einen NSC-Fehler zurück.

Schritt 6: Die NPI-Clients müssen beim Server-Host definiert werden

Wenn Sie den Natural Subprogram Server für die Verwendung eines externen Security-Systems konfigurieren (siehe *Natural API Server > Natural Subprogram Server auf z/OS > Konfiguration > Konfigurationsparameter > [SECURITY_MODE](#)*), müssen die NPI-Clients im externen Security-System definiert werden.

5

Konfiguration

■ Konfigurationserfordernisse	26
■ Konfiguration der externen Security	27
■ Konfigurationsdatei	27
■ Konfigurationsparameter	28
■ Beispiel für eine Konfigurationsdatei	34
■ Natural Subprogram Server Datasets	34

In diesem Dokument wird beschrieben, wie ein Natural Subprogram Server (NPI-Server) für z/OS (Batch) konfiguriert wird.

Dieses Dokument behandelt die folgenden Themen:

Konfigurationserfordernisse

Für einen Natural Subprogram Server for z/OS (Batch) ist folgende Konfiguration der z/OS-Language Environment-Parameter erforderlich:

Parameter	Definition
POSIX(ON)	Ermöglicht einem Natural Subprogram Server den Zugriff auf die POSIX-Funktionalität von z/OS. Wenn Sie einen Natural Subprogram Server mit POSIX(OFF) starten, wird er sofort mit einem User Abend vom Typ U4093 und der Systemmeldung EDC5167 beendet. IBM liefert die Standardeinstellung OFF.
TRAP(ON,NOSPIE)	Definiert die Behandlung von Programmabbrüchen in der LE/370-Umgebung: ON Aktiviert den Language Environment Condition Handler. NOSPIE Legt fest, dass das Language Environment Programmunterbrechungen und Programmabbrüche über einen ESTAE behandelt. Das bedeutet, dass der Natural Abend Handler für Programmabbrüche die Kontrolle erhält, um Programmunterbrechungen und Programmabbrüche zu behandeln. Wenn Sie TRAP(ON,NOSPIE) nicht angeben, funktioniert die Natural-Abend-Behandlung nicht ordnungsgemäß. IBM liefert die Standardeinstellung (ON,SPIE).
TERMTHDACT(UADUMP)	Legt fest, welche Informationen im Falle eines Abends ausgegeben werden. Die Option UADUMP erzeugt einen Sprachumgebungs-, einen CEEDUMP und einen System-Dump des Benutzeradressraums. Der CEEDUMP enthält keine für Natural relevanten Speicherbereiche. IBM liefert die Standardeinstellung (TRACE).
ENVAR(TZ=...)	Mit der Option ENVAR können Sie Linux-Umgebungsvariablen festlegen. Die einzige für den NPI-Server relevante Umgebungsvariable ist TZ (Zeitzone). Mit dieser Variablen können Sie den Zeitstempel in der Trace-Datei des NPI-Servers an Ihre Ortszeit anpassen. Beispiel (minus 1 Stunde Sommerzeit): ENVAR(TZ=CET-1DST) CET

Sie können die Parameter der z/OS Language Environment wie folgt festlegen:

- Durch Angabe des Parameters PARM auf der EXEC-Karte des Natural Subprogram Server-Startjobs. Die Länge der Optionen ist durch die maximale Länge des Parameters PARM begrenzt.

- Assemblieren Sie ein LE/370-Laufzeitoptionsmodul CEEUOPT und verlinken Sie es mit dem Natural Subprogram Server-Lademodul.

Konfiguration der externen Security

Wenn Sie den Natural Subprogram Server so konfigurieren, dass er die NPI-Clients im Natural Subprogram Server impersoniert (Konfigurationsparameter `SECURITY_MODE` auf `IMPERSONATE_LOCAL` gesetzt), muss der Natural Subprogram Server „programmgesteuert“ ausgeführt werden. Unter RACF sind für den Natural Subprogram Server folgende Definitionen erforderlich:

- Die Ressource `BPX.SERVER` muss definiert werden, und das Natural Subprogram Server-Konto muss über Lesezugriff (`READ`) auf diese Ressource verfügen.
- Die in der Startup-Job-Definition definierten `LOAD`-Datensätze müssen der Programmklasse `**` zugeordnet werden.

```
ralt program ** addmem('natural load library') uacc(read)
      ralt program ** addmem('NPI load library'//NOPADCHK) uacc(read)
      ralt program ** addmem('user load library'//NOPADCHK) uacc(read)
```

- `SETR WHEN(PROGRAM) REFRESH`

Zusätzlich muss jeder Client, der eine Verbindung zum Server herstellt, in RACF definiert werden und die Berechtigung zur Nutzung der z/OS Linux System Services erhalten.

Konfigurationsdatei

Eine Konfigurationsdatei wird dem Namen `STGCONFIG` zugeordnet.

Die Konfigurationsdatei ist eine Textdatei, die sich unter z/OS in einem Dataset oder in einer HFS-Datei befindet.

Die Konfigurationsdatei enthält die Serverkonfigurationsparameter in der Syntax `keyword=value` (Schlüsselwort=Wert). Darüber hinaus kann sie Kommentare enthalten, deren Anfang mit einem Rautezeichen (`#`).

Weitere Informationen siehe [Beispiel für eine Konfigurationsdatei](#).

Konfigurationsparameter

Die folgenden Konfigurationsparameter stehen zur Verfügung:

FRONTEND_NAME

Dieser Konfigurationsparameter gibt den Namen des Natural-Frontends an, das zum Starten einer Natural-Sitzung verwendet werden soll. Das Frontend befindet sich in einem PDS-Member. Es muss eintrittsinvariant (reentrant) oder wiederverwendbar verlinkt sein.

Wert	Erläuterung
frontend-name	Name des zu verwendenden Natural-Frontends. Die maximale Länge beträgt 8 Zeichen. Es ist kein Standardwert vorgegeben.

Beispiel:

```
FRONTEND_NAME=NATvrsSV
```

Dabei steht `vrs` für Versions-, Release- und System Maintenance Level-Nummer.

FRONTEND_OPTIONS

Die Werte dieses Konfigurationsparameters können verwendet werden, um zusätzliche Optionen für das Natural-Frontend festzulegen.

Wert	Erläuterung
40	Klonen nicht verwenden. Um die Initialisierung der Natural-Sitzung zu beschleunigen, wird jede neue Sitzung aus der bei der Serverinitialisierung erstellten Master-Sitzung geklont. Wenn Sie diese Option aktivieren, wird jede neue Sitzung von Grund auf neu initialisiert. Dies hat erhebliche Auswirkungen auf die Antwortzeit des Servers.

Beispiel:

```
FRONTEND_OPTIONS=40
```

FRONTEND_PARAMETER

Dieser optionale Konfigurationsparameter enthält zusätzliche Natural-Frontend-Parameter, wie im Bereich Startup-Parameter angegeben.

Wert	Erläuterung
parameter-name	Sie können mehrere Parameter definieren. Jede Parameterangabe besteht aus einem Paar von 8-stelligen Zeichenfolgen, wobei die erste das Parameterschlüsselwort und die zweite den Parameterwert enthält. Es ist kein Standardwert vorgegeben.

Beispiel:

```
FRONTEND_PARAMETER='MSGCLASSX      '
```

Die Einstellung in diesem Beispiel legt fest, dass die Standardausgabeklasse für CMPRINT X ist.

Weitere Informationen siehe *Operations (Verwaltung/Betrieb) > Natural als Server*.

HANDLE_ABEND

Tritt während der Serververarbeitung außerhalb der Natural-Verarbeitung ein Ablaufabbruch auf, wird dieser vom Natural Abend Handling nicht abgefangen. Aus diesem Grund verfügt der Natural Subprogram Server über eine eigene Abend Recovery.

Wert	Erläuterung
YES	Fängt Abbrüche bei der Serververarbeitung ab, erstellt einen Snap-Dump und bricht die Sitzung des betroffenen Benutzers ab. Dies ist der Standardwert. Es wird empfohlen, den Standardwert zu verwenden, um die Auswirkungen eines Abbruchs auf einen einzelnen Benutzer zu beschränken.
NO	Unterbindet die Server-Abbruchbehandlung. Jeder Abbruch bei der Serververarbeitung beendet die gesamte Serververarbeitung, was sich auf alle Benutzer dieses Servers auswirkt.

Beispiel:

```
HANDLE_ABEND=NO
```

HOST_NAME

Dieser optionale Konfigurationsparameter ist nur erforderlich, wenn der Server-Host mehrere TCP/IP-Stacks unterstützt.

Wert	Erläuterung
host_name	Wenn HOST_NAME angegeben ist, empfängt der Server Verbindungen über den durch HOST_NAME angegebenen Stack; andernfalls empfängt der Server Verbindungen über alle Stacks. Es ist kein Standardwert vorgegeben.

Beispiel:

```
HOST_NAME=node1
```

oder

```
HOST_NAME=157.189.160.55
```

IGNORE_PRESENT_SERVER

Ein Natural Subprogram Server ordnet eine sogenannte Serverumgebung zu, die die serverabhängigen gemeinsamen Ressourcen enthält. Diese Umgebung ist für jeden Server eindeutig und bezieht sich auf den Natural Subprogram Server-Namen.

Wenn ein Natural Subprogram Server nicht ordnungsgemäß beendet wird, kann dies dazu führen, dass die Serverumgebung in einem inkonsistenten Zustand verbleibt, wodurch nachfolgende Serverneustarts mit dem Fehler NAT9913 fehlschlagen.

Wenn Sie einen Natural Subprogram Server mit der Option `IGNORE_PRESENT_SERVER=YES` starten, kann dies zu einer Beschädigung eines bereits laufenden Servers führen, wenn dieser denselben Servernamen benutzt.

Wert	Erläuterung
YES	Beendet die vorhandene Serverumgebung.
NO	Bricht die Serverinitialisierung ab, wenn bereits eine Serverumgebung vorhanden ist. Dies ist der Standardwert.

Beispiel:

```
IGNORE_PRESENT_SERVER=YES
```

INITIAL_USERID

Bei der Serverinitialisierung erstellt der Natural Subprogram Server eine temporäre Natural-Sitzung, um die Properties der installierten Natural-Umgebung abzurufen.

Dieser Konfigurationsparameter legt die Benutzerkennung fest, die für diese Natural-Sitzung verwendet werden soll.

Wert	Erläuterung
userid	Der angegebene Wert darf nicht länger als 8 Zeichen sein, andernfalls wird er gekürzt.
SYSTEM	Dies ist der Standardwert.

Beispiel:

```
INITIAL_USERID=NPIINITU
```

Siehe auch *Natural API Server > Natural Subprogram Server auf z/OS > Installation > Installation des Natural Subprogram Server auf z/OS > [Schritt 5: Clients müssen in Natural Security definiert werden](#).*

PORT_NUMBER

Dieser Konfigurationsparameter legt die TCP/IP-Portnummer fest, über die eine Verbindung zum Server hergestellt werden kann.

Wert	Erläuterung
1 - 65535	TCP/IP-Portnummer. Es ist kein Standardwert vorgegeben.

Beispiel:

```
PORT_NUMBER=3140
```

SECURITY_MODE

Der Natural Subprogram Server bietet ein Sicherheitskonzept, das auch die Betriebssystemressourcen umfasst. Die Anmeldedaten des Clients werden im betriebssystemabhängigen Security-System überprüft, und die Anfrage des Clients wird gemäß den Kontodaten des Clients ausgeführt.

Mit dem Parameter `SECURITY_MODE` können Sie festlegen, ob Clients unter z/OS mit Impersonierung ausgeführt werden. In diesem Fall läuft die Subtask des Clients entsprechend dem Konto des Clients.

Wert	Erläuterung
IMPERSONATE_LOCAL	Die Impersonierung erfolgt innerhalb der Natural Subprogram Server-Umgebung. Der Client muss im externen Security-System des Natural Subprogram Servers (z. B. RACF) definiert sein. Es ist kein Standardwert vorgegeben.



Anmerkungen:

1. Der Parameter `SECURITY_MODE` setzt Natural Security (NSC) voraus.
2. Es gilt für Natural for Db2 (nur Batch Server): Um den Natural Subprogram Server mit aktivierter Impersonierung auszuführen, muss das Db2-Schnittstellenmodul `DSNRLI` (anstelle von `DSNALI`) mit dem Natural-Nukleus verlinkt sein.

Beispiel:

```
SECURITY_MODE=IMPERSONATE
```

SESSION_PARAMETER

Dieser optionale Konfigurationsparameter definiert dynamische Session-Parameter, die für alle auf dem Natural Subprogram Server ausgeführten Sitzungen gelten.

Wert	Erläuterung
parameter-string	Diese Zeichenfolge kann sich über mehrere Zeilen erstrecken. Ein Pluszeichen (+) am Ende einer Zeichenfolgezeile bedeutet, dass eine weitere Zeile folgt. Es ist kein Standardwert vorgegeben.

Beispiel:

```
SESSION_PARAMETER='NUCNAME=NATNUCvr,' +
'FNAT=(10,930),FSEC=(10,310),' +
'FUSER=(10,32),' +
'PROFILE=(NPIPARM,18006,48),ADAMODE=1,' +
'BPI=(TYPE=NAT,SIZE=6044)'
```

Dabei steht `vr` für die Versions- und Release-Nummer.

THREAD_NUMBER

Dieser Konfigurationsparameter legt die Anzahl der vom Natural-Frontend zuzuordnenden physischen Speicher-Threads fest, d. h. die Anzahl der Sitzungen, die parallel ausgeführt werden können.

Wert	Erläuterung
thread-number	Anzahl der zuzuordnenden physischen Speicher-Threads. Anmerkung: Diese Zahl begrenzt nicht die Anzahl der Sitzungen innerhalb des Servers, sondern die Anzahl der Sitzungen, die gleichzeitig im Ausführungsstatus sein können. Die Anzahl der Sitzungen wird durch die Größe des Natural-Auslagerungsmediums begrenzt.
3	Dies ist der Standardwert.

Beispiel:

```
THREAD_NUMBER=5
```

THREAD_SIZE

Dieser Konfigurationsparameter legt die Größe jedes physischen Speicher-Threads fest, der zur Ausführungszeit die Natural-Sitzungsdaten enthält.

Wert	Erläuterung
thread-size	Größe (in KB) jedes physischen Speicher-Thread.
500	Dies ist der Standardwert.

Beispiel:

```
THREAD_SIZE=800
```

TRACE_LEVEL

Wert	Erläuterung
trace-level	Siehe Abschnitt <i>Natural API Server > Natural Subprogram Server auf z/OS > Operation > Runtime Trace Facility > Trace Level</i> .
0	Dies ist der Standardwert.

Beispiel:

```
TRACE_LEVEL=0x00000011
```

oder

```
TRACE_LEVEL=31+27
```

Die Einstellung in diesem Beispiel schaltet die Bits 31 und 27 ein.

WORKER_DELAY

Dieser Konfigurationsparameter legt fest, wie lange (in Sekunden) ein Worker auf aufeinanderfolgende Anfragen wartet. Siehe auch [Worker Pool](#).

Wert	Erläuterung
worker-delay	Wert, der angibt, wie lange (in Sekunden) ein Worker auf nachfolgende Anfragen wartet.
0	Dies ist der Standardwert. Es wird kein Worker Pool verwendet.

Beispiel:

```
WORKER_DELAY=10
```

Beispiel für eine Konfigurationsdatei

```
# This is a comment
SESSION_PARAMETER= +
'profile=(npqa,10,930),' +
'fuser=(10,32)'
THREAD_NUMBER=2
THREAD_SIZE=700
FRONTEND_NAME=NATOS92L      # and another comment
PORT_NUMBER=4711
```

Natural Subprogram Server Datasets

Der Natural Subprogram Server benötigt die folgenden Datasets:

STGCONFIG	Definiert die Serverkonfigurationsdatei.
STGTRACE	Die Trace-Ausgabe des Servers.
STGSTDO	Der stdo-Dataset.
STGSTDE	Die stde-Fehlerausgabe.

6 Operation

■ Natural Subprogram Server starten	36
■ Natural Subprogram Server beenden	36
■ Natural Subprogram Server überwachen	37
■ Runtime Trace Facility	38

Dieses Dokument beschreibt, wie ein Natural Subprogram Server auf z/OS (Batch) betrieben wird.

Dieses Dokument behandelt die folgenden Themen:

Natural Subprogram Server starten

Der Natural Subprogram Server kann als eine Started Task gestartet werden:

```
//NPISRV  PROC
//KSPSRV  EXEC PGM=NATRNPI,REGION=4000K,TIME=1440,
// PARM=('POSIX(ON)/NPISRV1')
//STEPLIB DD DISP=SHR,DSN=NP1vrs.LOAD
//        DD DISP=SHR,DSN=NATvrs.LOAD
//CMPRINT DD SYSOUT=X
//STGCONFIG DD DISP=SHR,DSN=NP1vrs.CONFIG(SRV1)
//STGTRACE DD SYSOUT=X
//STGSTDO  DD SYSOUT=X
//STGSTDE  DD SYSOUT=X
```

Dabei steht `vrs` für Versions-, Release- und System Maintenance Level-Nummer des Natural Subprogram Server.

Die derzeit gültigen Versionen finden Sie unter <https://empower.softwareag.com/>.



Anmerkungen:

1. `PARM=('POSIX(ON)/NPISRV1')` - `POSIX(ON)` ist für eine ordnungsgemäße Initialisierung von LE370 erforderlich, wobei `NPISRV1` der Servername ist. Dieser Servername wird zur Referenzierung serverbezogener Ressourcen verwendet.
2. Der Name der Started Task muss unter RACF und den z/OS Linux System Services definiert werden.

Natural Subprogram Server beenden

Der Natural NPI Server kann mithilfe von z/OS-Operatorkommandos beendet werden.

`STOP NPISRV` stoppt den Server mit dem Namen der Started Task `NPISRV`.

`F NPISRV,APPL=TERMINATE` sendet das Kommando `TERMINATE` an die Started Task. Dadurch wird ebenfalls der Natural Subprogram Server angehalten. Weitere Informationen siehe Abschnitt [Monitor-Kommunikation](#).

Aktive Natural-Sitzungen werden beim Herunterfahren des Natural Subprogram Server abgebrochen.

Natural Subprogram Server überwachen

To enable the administrator to monitor the status of the Natural Subprogram Server, a monitor task is provided and is automatically initialized at server startup. Using the monitor commands described below, the administrator can control the server activities, cancel particular user sessions, terminate the entire server, etc.

Dieses Dokument behandelt die folgenden Themen:

- [Monitor-Kommunikation](#)
- [Monitor-Kommandos](#)

Monitor-Kommunikation

Zur Kommunikation mit dem Monitor können Sie das Operatorkommando `MODIFY` verwenden. Mit den Kommandos im Abschnitt [Monitor-Kommandos](#) lassen sich die Monitor-Kommandos ausführen. Die Ausgabe des ausgeführten Monitor-Kommandos wird in das Systemprotokoll geschrieben.

Beispiel:

```
F jobname,APPL=ping
```

sendet das Kommando `ping` an den Natural Subprogram Server, der unter dem Job `jobname` läuft.

Monitor-Kommandos

Der Natural Subprogram Server unterstützt die folgenden Monitor-Kommandos:

Monitor Command	Erläuterung
<code>ping</code>	Überprüft, ob der Server aktiv ist. Der Server antwortet und sendet die Zeichenfolge <code>I'm still up</code>
<code>terminate</code>	Beendet den Server.
<code>list sessions</code>	Für jede Sitzung gibt der Server Informationen über den Benutzer, dem die Sitzung gehört, den Zeitpunkt der Sitzungsinitialisierung, den Zeitpunkt der letzten Aktivität sowie eine interne Sitzungskennung (<code>session-id</code>) zurück.
<code>help</code>	Gibt Hilfeinformationen zu den unterstützten Monitorkommandos zurück.

Runtime Trace Facility

Zu Debugging-Zwecken enthält der Servercode eine eingebaute Nachverfolgungsfunktion, die aktiviert werden kann.

Dieses Dokument behandelt die folgenden Themen:

- [Trace-Aufzeichnungsmedium](#)
- [Trace-Konfiguration](#)
- [Trace Level](#)

Trace-Aufzeichnungsmedium

Ein entfernter Natural Subprogram Server schreibt sein Laufzeit-Trace in die logische Systemdatei `SYSOUT` der `FSIO`-Task.

Trace-Konfiguration

Die Nachverfolgung wird über einen Trace Level konfiguriert, der den Detaillierungsgrad der Trace-Informationen festlegt. Siehe auch [Trace Level](#).

Jede Natural-Sitzung verfügt über ein 32-Bit-Trace-Statuswort (TSW), das den Trace Level für diese Sitzung definiert. Der Wert des TSW wird im Konfigurationsparameter `TRACE_LEVEL` festgelegt. Der Wert Null bedeutet, dass der Trace deaktiviert ist.

Trace Level

Jedes Bit des TSW, beginnend mit dem Bit ganz rechts, ist für bestimmte Trace-Informationen zuständig:

Bit 31	Trace (Nachverfolgung) der wichtigsten Ereignisse (Server-Initialisierung/Beendigung, Client-Anfrage/Ergebnis).
Bit 30	Detail-Funktionen (Sitzungszuordnung, detaillierte Anfrageverarbeitung).
Bit 29	Speicherauszugerstellung der internen Speicherbereiche.
Bit 28	Zugriff auf das Sitzungsverzeichnis.
Bit 27	Frei.
Bit 26	Frei.
Bit 25	Frei.
Bit 24	Frei.
Bit 23	Callback-Anfragen (Natural-Laufzeitumgebung ruft Frontend Stub auf).
Bit 22	Callback-Speicherfunktionen (Natural-Laufzeitumgebung ruft Frontend Stub wegen Speicherzuordnung auf).

Bit 21	Trace (Nachverfolgung) der NSC-Anmeldeverarbeitung.
Bit 20	Ereignisse im Worker Pool (Hinzufügen/Löschen/Buchen eines Workers).
Bit 19	Der Worker Pool-Inhalt wird bei jedem Hinzufügen oder Löschen eines Workers protokolliert.
Bit 18-00	Frei.

7

Sicherheit

■ Natural Security	42
■ Externe Security	42

Dieses Dokument beschreibt die Security-Umgebung eines Natural Subprogram Servers unter z/OS (Batch).

Dieses Dokument behandelt die folgenden Themen:

Natural Security

Jede Anfrage enthält die Benutzerkennung und das Benutzerpasswort. Der Natural Subprogram Server führt mit diesen Anmeldedaten eine Anmeldung (LOGON) durch. Natural Security überprüft die Benutzerkennung und das Passwort. Alle für diesen Benutzer in Natural Security festgelegten Einschränkungen werden wirksam.

Aus Sicht des Betriebssystems arbeitet der Benutzer anonym über das Konto des Natural Subprogram Server.

Externe Security

Der Natural Subprogram Server kann so konfiguriert werden, dass er mit Impersonierung ausgeführt wird. Weitere Informationen siehe *Natural API Server > Natural Subprogram Server auf z/OS > Konfiguration > Konfigurationsparameter > [SECURITY-MODE](#)*.

Wenn der Server für den Betrieb mit Impersonierung konfiguriert ist, wird jeder Client vom z/OS Security System validiert. Die z/OS-Subtask des Clients läuft über das Konto des Clients. Alle für diesen Benutzer in RACF festgelegten Einschränkungen werden wirksam. Jeder Client muss im z/OS-Sicherheitssystem definiert sein. Weitere Informationen siehe *Natural API Server > Natural Subprogram Server auf z/OS > Konfiguration > [Externe Security-Konfiguration](#)*.

Für die Verwendung der Impersonierung wird Natural Security (NSC) vorausgesetzt. Wird ein Benutzer vom z/OS Security System validiert, wird er von NSC als vertrauenswürdiger (trusted) Benutzer akzeptiert. Das heißt, der Benutzer muss in NSC definiert sein, es wird jedoch bei der Anmeldung kein Passwort überprüft. Daher sind auch bei der Impersonierung alle für diesen Benutzer in Natural Security festgelegten Einschränkungen wirksam.

III

Natural OpenAPI Server auf Tomcat

Dieses Dokument behandelt die folgenden Themen:

Installation
Konfiguration
Betrieb
Überwachung
Sicherheit

8 Installation

■ Installation des Natural OpenAPI Server	46
---	----

In diesem Dokument wird beschrieben, wie der Natural OpenAPI Server installiert wird.

Installation des Natural OpenAPI Server

➤ Um den Natural OpenAPI Server zu installieren:

- 1 Öffnen Sie den Software AG Installer und navigieren Sie zum Installationsverzeichnis.
- 2 Erweitern Sie den Knoten **Natural** und wählen Sie **Natural OpenAPI Server**.
- 3 Befolgen Sie die Anleitung auf dem Bildschirm, um die Installation fertigzustellen.

9 Konfiguration

■ Allgemeine Konfiguration	48
■ Mapping von Natural-Fehlern auf HTTP-Statuscodes	49

Dieses Dokument behandelt die folgenden Themen:

Allgemeine Konfiguration

Die folgenden Konfigurationseinstellungen müssen vor dem Ausführen des Natural OpenAPI Server vorgenommen werden.

Definieren Sie in der Datei *application.properties* die folgenden Properties:

Parameter	Beschreibung
<code>com.softwareag.natural.openapi.server.config.host.address</code>	Die IP-Adresse des Natural Subprogram Servers.
<code>com.softwareag.natural.openapi.server.config.host.port</code>	Der Port für die Verbindung zum Natural Subprogram Server.
<code>natural.integration.services.files</code>	Eine Liste von JSON-Dateien, auf die der OpenAPI Server zugreifen können soll (Dateinamen, durch Kommas getrennt). Beispiel: <code>/opt/softwareag/NaturalOpenAPIServer/services/1.json</code> <code>/opt/softwareag/NaturalOpenAPIServer/services/2.json</code>
<code>natural.integration.services.directory</code>	Der Speicherort des Ordners, der alle Dienstdefinitionsdateien enthält.
<code>com.softwareag.natural.openapi.server.config.host.ssl.enabled</code>	Legen Sie fest, ob SSL verwendet werden soll.
<code>com.softwareag.natural.openapi.server.config.host.ssl.certsChain[0]</code>	Speicherort der öffentlichen Zertifikatsdatei für die Verbindung zum Natural Subprogram Server über SSL-Kommunikation.
<code>com.softwareag.natural.openapi.web.config.authenticationType</code>	Die Authentifizierungsmethode für die Verbindung zum Natural Subprogram Server.
<code>com.softwareag.natural.openapi.web.config.oAuth2Config.tokenUrl</code>	Die Token-URL des Credential-Providers, um ein Bearer-Token beim Testen mit dem Swagger-UI-Web-Client abzurufen.
<code>com.softwareag.natural.openapi.web.config.oAuth2Config.scopes</code>	Die Bereiche, die beim Testen mit dem Swagger-UI-Web-Client vom Credential-Provider für ein Bearer-Token angefordert werden sollen.
<code>com.softwareag.natural.openapi.server.config.host.charset</code>	Der Zeichensatz, der bei der Kodierung der Anfragen an den Natural Subprogram Server verwendet werden soll.
<code>management.endpoints.web.exposure.include</code>	Dieser Parameter steuert die Bereitstellung des Endpunkts des Reload-Dienstes. Ist er nicht definiert, muss diese Property mit dem Wert <code>reload</code> definiert und hinzugefügt werden. Ist er bereits definiert, muss er nicht hinzugefügt werden.

Parameter	Beschreibung
	reloadServices an die vorhandene Services-Liste an. getrennten Werte an.



Anmerkungen:

1. Sie können eine Dateiliste in der Property `natural.integration.services.files` und einen weiteren Ordner in der Property `natural.integration.services.directory` definieren. Die Anwendung nutzt die Ressourcen aus beiden Speicherorten.
2. Alle vom Natural OpenAPI Eclipse-Plug-in erstellten JSON-Dateien sollten in den Service-Ordner kopiert werden (der in der Datei `application.properties` durch die Property `natural.integration.services.directory` definiert ist oder in der Property `natural.integration.services.files` angegeben werden sollte).

Mapping von Natural-Fehlern auf HTTP-Statuscodes

Der Natural OpenAPI Server kann Natural-Fehlercodes aus dem Backend auf HTTP-Statuscodes mappen.

Die Reihenfolge der Zuordnung lautet wie folgt:

1. Exakte Übereinstimmung (z. B. NAT3009)
2. Übereinstimmung mit einem numerischen Bereich (z. B. 0..99)
3. Globales Standard-Mapping

Standard-Mappings sind im Natural OpenAPI Server vordefiniert.

Sie können Mappings in der Datei `application.properties` mithilfe des Präfixes `error-mapping.*` überschreiben oder erweitern.

Beispiel: Einen Natural-spezifischen Code überschreiben

```
error-mapping.sources.natural.specific.NAT3009.http-code=422
error-mapping.sources.natural.specific.NAT3009.http-message=Unprocessable Entity
error-mapping.sources.natural.specific.NAT3009.category=VALIDATION_ERROR
error-mapping.sources.natural.specific.NAT3009.log-level=WARN
```

Beispiel: Ein Natural-Range-Mapping definieren

```
error-mapping.sources.natural.ranges.syntax.from=0
error-mapping.sources.natural.ranges.syntax.to=49
error-mapping.sources.natural.ranges.syntax.http-code=400
error-mapping.sources.natural.ranges.syntax.http-message=Bad Request
error-mapping.sources.natural.ranges.syntax.category=SYNTAX_ERROR
error-mapping.sources.natural.ranges.syntax.log-level=WARN
```

Beispiel: Globales Fallback-Mapping überschreiben

```
error-mapping.default-mapping.http-code=500
error-mapping.default-mapping.http-message=Internal Server Error
error-mapping.default-mapping.category=SYSTEM_ERROR
error-mapping.default-mapping.log-level=ERROR
```

Zusätzlich zu den Natural-Code-Mappings werden Transportfehlschläge durch den WebFlux Error Handler zugeordnet:

- Fehlschlag der Backend-Verbindung: 502 Bad Gateway
- Backend-Zeitüberschreitung: 504 Gateway Timeout
- Unbekannte/nicht gemappte Fehler: 500 Internal Server Error

10

Betrieb

- Bereitstellung der Webanwendung mit Tomcat 52
- Bereitstellung von OpenAPI-Ressourcen 52

Dieses Dokument behandelt die folgenden Themen:

Bereitstellung der Webanwendung mit Tomcat

Im Folgenden finden Sie einige spezifische Konfigurationen für Apache Tomcat®, falls Sie die Natural OpenAPI-Webanwendung mit Ihrem eigenen Tomcat bereitstellen möchten.

- [Bereitstellung der War-Datei](#)
- [Konfiguration der Datei Server.XML](#)

Bereitstellung der War-Datei

Die Datei *NaturalOpenAPIServer.war* muss in das Verzeichnis */tomcat/webapps/* kopiert werden.

Zusätzlich muss die Umgebungsvariable `SPRING_CONFIG_LOCATION` so konfiguriert werden, dass sie auf den Speicherort der Datei *application.properties* zeigt.

Konfiguration der Datei Server.XML

Alternativ kann die folgende Konfiguration bei der Bereitstellung des Natural OpenAPI-Servers zur Datei *server.xml* von Tomcat hinzugefügt werden.

```
<Context path="/noa1" docBase="noa1" override="true">
  <Environment name="SPRING_CONFIG_LOCATION" ↵
value="/your-path/conf/application.properties" type="java.lang.String" />
</Context>
```

Bereitstellung von OpenAPI-Ressourcen

Um die im Abschnitt *Developer Overview* > [Generating OpenAPI Resources](#) beschriebenen Dienste bereitzustellen, muss der Natural OpenAPI Server mit dem Speicherort der Dienstdateien konfiguriert werden.

Der Speicherort der Dienstdateien kann auf folgende Weise konfiguriert werden:

- Unter Verwendung von `natural.integration.services.directory` – Legen Sie diese Property in der Datei *application.properties* fest, um das Verzeichnis anzugeben, das die Dienstdateien enthält. Speichern Sie die Dienstdateien in dem angegebenen Verzeichnis.
- Unter Verwendung von `natural.integration.services.files` – Legen Sie diese Property fest, um die Namen der spezifischen Dienstdateien anzugeben, die bereitgestellt werden sollen. Die angegebenen Dateien müssen sich im Class-Pfad der Anwendung befinden.

Beim Start des Natural OpenAPI-Servers werden die konfigurierten Dienstdateien geladen und die darin definierten OpenAPI-Dienste registriert.

Jede Dienstdatei enthält die Definitionen und Mapping-Informationen, die erforderlich sind, um ein Natural-Subprogramm als REST API-Dienst bereitzustellen. Sobald der Server erfolgreich gestartet wurde, stehen alle in den geladenen Dienstdateien definierten Dienste den Clients über den Natural OpenAPI Server zur Verfügung. Weitere Informationen finden Sie unter dem Parameter `management.endpoints.web.exposure.include` im Abschnitt [Allgemeine Konfiguration](#) section.

Um einen Dienst hinzuzufügen oder zu ändern, muss daher die entsprechende Dienstdatei aktualisiert und der Server neu gestartet werden oder der Reload-Aktuator-Endpunkt verwendet werden, damit die Änderungen geladen werden und die Dienste verfügbar sind.

11 Überwachung

OpenTelemetry ist ein offener Standard zur Erfassung und Verwaltung von beobachtbaren Daten aus Anwendungen und Diensten. Er bietet ein gemeinsames Framework, das dabei hilft, das Verhalten, die Leistung und den Gesamtzustand eines Systems zu überwachen und zu analysieren.

Unser Produkt nutzt die drei von OpenTelemetry definierten Haupttypen von Telemetriedaten:

- **Traces**

Nachverfolgungen liefern Informationen über den Datenfluss einer Anfrage durch das System. Traces helfen dabei, einen Vorgang über verschiedene Komponenten hinweg zu verfolgen und festzustellen, wo Zeit verloren geht oder wo Fehler auftreten.

- **Metrics**

Metriken liefern numerische Messwerte zum System, wie beispielsweise Ressourcenauslastung, Anzahl der Anfragen, Antwortzeiten und andere leistungsbezogene Werte. Metriken können zur Überwachung des Gesamtverhaltens und -zustands der Anwendung verwendet werden.

- **Logs**

Protokolle liefern detaillierte Aufzeichnungen von Ereignissen, die während der Ausführung der Anwendung auftreten. Protokolle können dabei helfen, Fehler, Warnungen und andere relevante Laufzeitereignisse zu untersuchen.

Durch die Erfassung von Nachverfolgungen, Metriken und Protokollen bietet OpenTelemetry einen umfassenderen Überblick über das System und erleichtert die Überwachung des Anwendungsverhaltens, die Identifizierung von Leistungsproblemen sowie die Fehlerbehebung.

12

Sicherheit

■ Sichere Kommunikation zwischen dem Natural OpenAPI Server und dem Subprogram Server	58
■ Sichere Kommunikation zwischen Client und Natural OpenAPI Server	59
■ Authentifizierung	60

Der Natural OpenAPI Server bietet Konfigurationsoptionen für die Sicherheit der Kommunikation:

- zwischen dem Natural OpenAPI Server und dem Subprogram Server.
- zwischen dem Client und dem Natural OpenAPI Server.

Dieses Dokument behandelt die folgenden Themen:

Sichere Kommunikation zwischen dem Natural OpenAPI Server und dem Subprogram Server

Mit dem Natural OpenAPI Server kann die Kommunikation zwischen dem Natural OpenAPI Server und den Subprogram Servers mittels SSL/TLS verschlüsselt werden.

SSL/TLS kann über die folgende Property aktiviert werden:

```
com.softwareag.natural.openapi.server.config.host.ssl.enabled=true
```

Wenn SSL/TLS eingerichtet ist, verwendet der Server die konfigurierte Zertifikatskette, um sichere Verbindungen herzustellen. Das Zertifikat kann über die Property `certsChain` konfiguriert werden:

```
com.softwareag.natural.openapi.server.config.host.ssl.certsChain[0]=classpath:natsubsrv.server.cert.crt
```

Die Property gibt den Speicherort des serverseitigen Zertifikats an. Das Präfix `classpath:` bedeutet, dass das Zertifikat aus dem Class-Pfad der Anwendung geladen wird.

Es können mehrere Zertifikate als Teil der Zertifikatskette angegeben werden, indem zusätzliche indizierte Properties verwendet werden, zum Beispiel: `certsChain[1]`, `certsChain[2]` und so weiter.



Anmerkung: Die SSL-Zertifikatskonfiguration ist nur relevant, wenn

`com.softwareag.natural.openapi.server.config.host.ssl.enabled` auf `true` gesetzt ist.

Sichere Kommunikation zwischen Client und Natural OpenAPI Server

Der Natural OpenAPI Server unterstützt sichere Kommunikation über SSL/TLS. Wenn SSL/TLS aktiviert ist, stellen Clients die Verbindung zum Server über HTTPS statt über HTTP her.

Der vom Server standardmäßig verwendete Port ist 8080. Für HTTPS kann ein anderer Port konfiguriert werden, beispielsweise 8443, indem die folgende Property festgelegt wird:

```
server.port=8443
```

Um SSL/TLS zu aktivieren, setzen Sie die folgende Property:

```
server.ssl.enabled=true
```

Die folgenden Properties konfigurieren das vom Server verwendete SSL-Zertifikat und den Schlüsselspeicher (Key Store):

Pfad zum Schlüsselspeicher

Diese Property gibt den Pfad zum Schlüsselspeicher an, der das SSL-Zertifikat und den privaten Schlüssel enthält. Beispiel:

```
server.ssl.key-store=file:///opt/softwareag/NaturalOpenAPIServer/conf/keystore.p12
```

Passwort für den Schlüsselspeicher

Diese Property gibt das Passwort an, das für den Zugriff auf den Schlüsselspeicher erforderlich ist. Beispiel:

```
server.ssl.key-store-password=password
```

Typ des Schlüsselspeichers

Diese Property gibt den Typ des Schlüsselspeichers an. Die unterstützten Typen sind JKS und PKCS12. Beispiel:

```
server.ssl.key-store-type=PKCS12
```

Schlüsselspeicher-Alias

Diese Property gibt den Alias an, der zur Identifizierung des Zertifikats und des privaten Schlüssels innerhalb des Schlüsselspeichers verwendet wird. Beispiel::

```
server.ssl.key-alias=noa-server
```



Anmerkung: Der Schlüsselspeicher muss ein gültiges SSL/TLS-Zertifikat und den dazugehörigen privaten Schlüssel enthalten.

Authentifizierung

Der Natural OpenAPI Server authentifiziert den Client, der die Anfrage sendet, nicht selbst, sondern stützt sich auf das Authentifizierungsmodul des Natural Subprogram Server. Ist der Natural Subprogram Server jedoch für die Verwendung einer Authentifizierungsmethode konfiguriert, müssen Sie die entsprechende Konfiguration im Natural OpenAPI Server festlegen. Durch die Festlegung der korrekten Konfiguration wird die entsprechende Definition im OpenAPI-Dokument unter `http://localhost:OpenAPIServer_port/v3/api-docs` erstellt.

Es gibt drei verschiedene Möglichkeiten, die Authentifizierung zu definieren:

- [Verwendung der Basic-Authentifizierung](#)
- [Verwendung eines OAuth2-Tokens](#)
- [Verwendung anonymer Anfragen](#)

Verwendung der Basic-Authentifizierung

Wird die Property `com.softwareag.natural.openapi.web.config.authenticationType` auf `basic` gesetzt, muss die Client-Anwendung Benutzername und Passwort in Base64-Kodierung (definiert in RFC-7617) übermitteln.

Verwendung eines OAuth2-Tokens

Das Abrufen des Tokens obliegt der Client-Anwendung, während die Token-Validierung vom Authentifizierungsmodul des Natural Subprogram Server durchgeführt wird.

Wird die Property `com.softwareag.natural.openapi.web.config.authenticationType` auf `oauth2` gesetzt, muss die Client-Anwendung gemäß den Anforderungen des OpenAPI-Dokuments ein Bearer-Token senden.

Die Client-Anwendung muss neben dem bereitgestellten Token den speziellen HTTP-Header `X-Impersonate-User` einfügen, um den zu impersonierenden Benutzer anzugeben.

Die folgende Konfiguration kann beim Testen mit dem Swagger-UI-Webclient verwendet werden. Das in den Natural OpenAPI Server eingebettete Swagger-UI unterstützt derzeit nur den Client Credentials-Ablauf. Wenn die OAuth2-Authentifizierung aktiviert ist, können Sie die `tokenURL` und die erforderlichen Bereiche definieren. Diese Werte werden im OpenAPI-Dokument angegeben, aber vom Natural OpenAPI Server nicht zur Authentifizierung verwendet. Im Falle einer Impersonierung muss diese nach einer erfolgreichen Token-Überprüfung durchgeführt werden.

Verwendung anonymer Anfragen

Wenn in der Datei *application.properties* kein Authentifizierungstyp definiert ist, gelten die Anfragen standardmäßig als anonym und sollten nicht authentifiziert werden. Es ist auch möglich, die Anfrage unter Verwendung des speziellen HTTP-Header *X-Impersonate-User* als anderer Benutzer auszuführen.

IV

Natural OpenAPI Server auf Liberty

Dieses Dokument behandelt die folgenden Themen:

Installation
Konfiguration
Betrieb
Überwachung
Sicherheit

13

Installation

■ Installation des Natural OpenAPI Server	66
---	----

Dieses Dokument behandelt die folgenden Themen:

Installation des Natural OpenAPI Server

➤ Um den Natural OpenAPI Server zu installieren:

- 1 Öffnen Sie den Software AG Installer und navigieren Sie zum Installationsverzeichnis.
- 2 Erweitern Sie den Knoten **Natural** und wählen Sie **Natural OpenAPI Server**.
- 3 Befolgen Sie die Anleitung auf dem Bildschirm, um die Installation fertigzustellen.

14

Konfiguration

■ Allgemeine Konfiguration	68
■ Mapping von Natural-Fehlern als HTTP-Statuscodes	69

Dieses Dokument behandelt die folgenden Themen:

Allgemeine Konfiguration

Die folgenden Konfigurationseinstellungen müssen vor dem Ausführen des Natural OpenAPI Server vorgenommen werden.

Definieren Sie in der Datei *application.properties* die folgenden Properties:

Parameter	Beschreibung
<code>com.softwareag.natural.openapi.server.config.host.address</code>	Die IP-Adresse des Natural Subprogram Server.
<code>com.softwareag.natural.openapi.server.config.host.port</code>	Der Port für die Verbindung zum Natural Subprogram Server.
<code>natural.integration.services.files</code>	Eine Liste von JSON-Dateien, auf die der OpenAPI Server zugreifen können soll (Dateinamen, durch Kommas getrennt). Beispiel: <code>/opt/softwareag/NaturalOpenAPIServer/services/</code> <code>/opt/softwareag/NaturalOpenAPIServer/services/</code>
<code>natural.integration.services.directory</code>	Der Speicherort des Ordners, der alle Dienstdefinitionsdateien enthält.
<code>com.softwareag.natural.openapi.server.config.host.ssl.enabled</code>	Legen Sie fest, ob SSL verwendet werden soll.
<code>com.softwareag.natural.openapi.server.config.host.ssl.certsChain[0]</code>	Speicherort der öffentlichen Zertifikatsdatei für die Verbindung zum Natural Subprogram Server über SSL-Kommunikation.
<code>com.softwareag.natural.openapi.web.config.authenticationType</code>	Die Authentifizierungsmethode für die Verbindung zum Natural Subprogram Server.
<code>com.softwareag.natural.openapi.web.config.oAuth2Config.tokenUrl</code>	Die Token-URL des Credential-Providers, um ein Bearer-Token beim Testen mit dem Swagger-UI-Web-Client abzurufen.
<code>com.softwareag.natural.openapi.web.config.oAuth2Config.scopes</code>	Die Bereiche, die beim Testen mit dem Swagger-UI-Web-Client vom Credential-Provider für ein Bearer-Token angefordert werden sollen.
<code>com.softwareag.natural.openapi.server.config.host.charset</code>	Der Zeichensatz, der bei der Kodierung der Anfragen an den Natural Subprogram Server verwendet werden soll.
<code>management.endpoints.web.exposure.include</code>	Dieser Parameter steuert die Bereitstellung des Endpunkts des Reload-Dienstes. Ist er nicht definiert, muss diese Property mit dem Wert <code>reload</code> definiert und hinzugefügt werden. Ist er bereits definiert, muss er nicht hinzugefügt werden.

Parameter	Beschreibung
	<code>reloadServices</code> an die vorhandene Konfiguration an die vorhandenen getrennten Werte an.



Anmerkungen:

1. Sie können eine Dateiliste in der Property `natural.integration.services.files` und einen weiteren Ordner in der Property `natural.integration.services.directory` definieren. Die Anwendung nutzt die Ressourcen aus beiden Speicherorten.
2. Alle vom Natural OpenAPI Eclipse-Plug-in erstellten JSON-Dateien sollten in den Service-Ordner kopiert werden (der in der Datei `application.properties` durch die Property `natural.integration.services.directory` definiert ist oder in der Property `natural.integration.services.files` angegeben werden sollte).

Mapping von Natural-Fehlern als HTTP-Statuscodes

Der Natural OpenAPI Server kann Natural-Fehlercodes aus dem Backend als HTTP-Statuscodes mappen.

Die Reihenfolge der Zuordnung lautet wie folgt:

1. Exakte Übereinstimmung (z. B. NAT3009)
2. Übereinstimmung mit einem numerischen Bereich (z. B. 0..99)
3. Globales Standard-Mapping

Standard-Mappings sind im Natural OpenAPI Server vordefiniert.

Sie können Mappings in der Datei `application.properties` mithilfe des Präfixes `error-mapping.*` überschreiben oder erweitern.

Beispiel: Einen Natural-spezifischen Code überschreiben

```
error-mapping.sources.natural.specific.NAT3009.http-code=422
error-mapping.sources.natural.specific.NAT3009.http-message=Unprocessable Entity
error-mapping.sources.natural.specific.NAT3009.category=VALIDATION_ERROR
error-mapping.sources.natural.specific.NAT3009.log-level=WARN
```

Beispiel: Ein Natural-Range-Mapping definieren

```
error-mapping.sources.natural.ranges.syntax.from=0
error-mapping.sources.natural.ranges.syntax.to=49
error-mapping.sources.natural.ranges.syntax.http-code=400
error-mapping.sources.natural.ranges.syntax.http-message=Bad Request
error-mapping.sources.natural.ranges.syntax.category=SYNTAX_ERROR
error-mapping.sources.natural.ranges.syntax.log-level=WARN
```

Beispiel: Globales Fallback-Mapping überschreiben

```
error-mapping.default-mapping.http-code=500
error-mapping.default-mapping.http-message=Internal Server Error
error-mapping.default-mapping.category=SYSTEM_ERROR
error-mapping.default-mapping.log-level=ERROR
```

Zusätzlich zu den Natural-Code-Mappings werden Transportfehlschläge durch den WebFlux Error Handler zugeordnet:

- Fehlschlag der Backend-Verbindung: 502 Bad Gateway
- Backend-Zeitüberschreitung: 504 Gateway Timeout
- Unbekannte/nicht zugeordnete Fehler: 500 Internal Server Error

15

Betrieb

■ Bereitstellung der Webanwendung mit Liberty	72
■ Bereitstellung von OpenAPI-Ressourcen	73

Dieses Dokument behandelt die folgenden Themen:

Bereitstellung der Webanwendung mit Liberty

Im Folgenden finden Sie einige spezifische Konfigurationen für Liberty, falls Sie die Natural OpenAPI-Webanwendung mit Ihrem eigenen Liberty bereitstellen möchten.

Bereitstellung der War-Datei

Die Datei *NaturalOpenAPIServer.war* muss in das Verzeichnis */config/apps/* kopiert werden.

Konfiguration der Datei *Server.XML*

Die folgende Konfiguration sollte zur Datei *server.xml* hinzugefügt werden, um den Natural OpenAPI Server bereitzustellen.

Zusätzlich muss die Umgebungsvariable `SPRING_CONFIG_LOCATION` konfiguriert werden, damit sie auf den Speicherort der Datei *application.properties* zeigt.

```
<?xml version="1.0" encoding="UTF-8"?>
  <server description="Natural _OpenAPI">

    <featureManager>
      <feature>servlet-6.0</feature>
      <feature>mpConfig-3.1</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint" httpPort="9080" ↵
httpsPort="9443"/>

    <!--
      appProperties is Liberty's per-application config mechanism: values ↵
here
      are scoped strictly to the <webApplication> that declares them and are
exposed through MicroProfile Config - MpConfigEnvironmentInitializer ↵
in
      the app reads them from there.
    -->
    <webApplication id="app1" name="app1" ↵
location="/config/apps/NaturalOpenAPIServer.war" contextRoot="/app1">
      <appProperties>
        <property name="SPRING_CONFIG_LOCATION" ↵
value="file:///config/application.properties"/>
      </appProperties>
    </webApplication>
  </server>
```

Bereitstellung von OpenAPI-Ressourcen

Um die im Abschnitt *Developer Overview* > [Generating OpenAPI Resources](#) beschriebenen Dienste bereitzustellen, muss der Natural OpenAPI Server mit dem Speicherort der Dienstdateien konfiguriert werden.

Der Speicherort der Dienstdateien kann auf folgende Weise konfiguriert werden:

- Unter Verwendung von `natural.integration.services.directory` – Legen Sie diese Property in der Datei `application.properties` fest, um das Verzeichnis anzugeben, das die Dienstdateien enthält. Speichern Sie die Dienstdateien in dem angegebenen Verzeichnis.
- Unter Verwendung von `natural.integration.services.files` – Legen Sie diese Property fest, um die Namen der spezifischen Dienstdateien anzugeben, die bereitgestellt werden sollen. Die angegebenen Dateien müssen sich im Class-Pfad der Anwendung befinden.

Beim Start des Natural OpenAPI-Servers werden die konfigurierten Dienstdateien geladen und die darin definierten OpenAPI-Dienste registriert.

Jede Dienstdatei enthält die Definitionen und Mapping-Informationen, die erforderlich sind, um ein Natural-Subprogramm als REST API-Dienst bereitzustellen. Sobald der Server erfolgreich gestartet wurde, stehen alle in den geladenen Dienstdateien definierten Dienste den Clients über den Natural OpenAPI Server zur Verfügung. Weitere Informationen finden Sie unter dem Parameter `management.endpoints.web.exposure.include` im Abschnitt [Allgemeine Konfiguration](#).

Um einen Dienst hinzuzufügen oder zu ändern, muss daher die entsprechende Dienstdatei aktualisiert und der Server neu gestartet werden oder der Reload-Actuator-Endpunkt verwendet werden, um die Änderungen zu laden und die Dienste verfügbar zu machen.

16 Überwachung

OpenTelemetry ist ein offener Standard zur Erfassung und Verwaltung von beobachtbaren Daten aus Anwendungen und Diensten. Er bietet ein gemeinsames Framework, das dabei hilft, das Verhalten, die Leistung und den Gesamtzustand eines Systems zu überwachen und zu analysieren.

Unser Produkt nutzt die drei von OpenTelemetry definierten Haupttypen von Telemetriedaten:

- **Traces**

Traces, d.h. Nachverfolgungen, liefern Informationen über den Datenfluss einer Anfrage durch das System. Traces helfen dabei, einen Vorgang über verschiedene Komponenten hinweg zu verfolgen und festzustellen, wo Zeit verbraucht wird oder wo Fehler auftreten.

- **Metrics**

Metriken liefern numerische Messwerte zum System, beispielsweise Ressourcenauslastung, Anzahl der Anfragen, Antwortzeiten und andere leistungsbezogene Werte. Metriken können zur Überwachung des Gesamtverhaltens und -zustands der Anwendung verwendet werden.

- **Logs**

Logs, d.h. Protokolle, liefern detaillierte Aufzeichnungen von Ereignissen, die während der Ausführung der Anwendung auftreten. Protokolle können dabei helfen, Fehler, Warnungen und andere relevante Laufzeitereignisse zu untersuchen.

Durch die Erfassung von Nachverfolgungen, Metriken und Protokollen bietet OpenTelemetry einen umfassenderen Überblick über das System und erleichtert die Überwachung des Anwendungsverhaltens, die Identifizierung von Leistungsproblemen sowie die Fehlerbehebung.

17

Sicherheit

■ Sichere Kommunikation zwischen dem Natural OpenAPI Server und dem Subprogram Server	78
■ Sichere Kommunikation zwischen Client und Natural OpenAPI Server	79
■ Authentifizierung	80

Der Natural OpenAPI Server bietet Konfigurationsoptionen für die Sicherheit der Kommunikation:

- zwischen dem Natural OpenAPI Server und dem Subprogram Server.
- zwischen dem Client und dem Natural OpenAPI Server.

Dieses Dokument behandelt die folgenden Themen:

Sichere Kommunikation zwischen dem Natural OpenAPI Server und dem Subprogram Server

Mit dem Natural OpenAPI Server kann die Kommunikation zwischen dem Natural OpenAPI-Server und den Subprogram Servers mittels SSL/TLS verschlüsselt werden.

SSL/TLS kann über die folgende Property aktiviert werden:

```
com.softwareag.natural.openapi.server.config.host.ssl.enabled=true
```

Wenn SSL/TLS eingerichtet ist, verwendet der Server die konfigurierte Zertifikatskette, um sichere Verbindungen herzustellen. Das Zertifikat kann über die Property `certsChain` konfiguriert werden:

```
com.softwareag.natural.openapi.server.config.host.ssl.certsChain[0]=classpath:natsubsrv.server.cert.crt
```

Die Property gibt den Speicherort des serverseitigen Zertifikats an. Das Präfix `classpath:` bedeutet, dass das Zertifikat aus dem Class-Pfad der Anwendung geladen wird.

Es können mehrere Zertifikate als Teil der Zertifikatskette angegeben werden, indem zusätzliche indizierte Properties verwendet werden, zum Beispiel: `certsChain[1]`, `certsChain[2]` und so weiter.



Anmerkung: Die SSL-Zertifikatskonfiguration ist nur relevant, wenn

`com.softwareag.natural.openapi.server.config.host.ssl.enabled` auf `true` gesetzt ist.

Sichere Kommunikation zwischen Client und Natural OpenAPI Server

Der Natural OpenAPI Server unterstützt sichere Kommunikation über SSL/TLS. Wenn SSL/TLS aktiviert ist, stellen Clients die Verbindung zum Server über HTTPS statt über HTTP her.

Der vom Server standardmäßig verwendete Port ist 8080. Für HTTPS kann ein anderer Port konfiguriert werden, beispielsweise 8443, indem die folgende Property festgelegt wird:

```
server.port=8443
```

Um SSL/TLS zu aktivieren, setzen Sie die folgende Property:

```
server.ssl.enabled=true
```

Die folgenden Properties konfigurieren das vom Server verwendete SSL-Zertifikat und den Schlüsselspeicher (Key Store):

Pfad zum Schlüsselspeicher

Diese Property gibt den Pfad zum Schlüsselspeicher an, der das SSL-Zertifikat und den privaten Schlüssel enthält. Beispiel:

```
server.ssl.key-store=file:///opt/softwareag/NaturalOpenAPIServer/conf/keystore.p12
```

Passwort für den Schlüsselspeicher

Diese Property gibt das Passwort an, das für den Zugriff auf den Schlüsselspeicher erforderlich ist. Beispiel:

```
server.ssl.key-store-password=password
```

Typ des Schlüsselspeichers

Diese Property gibt den Typ des Schlüsselspeichers an. Die unterstützten Typen sind JKS und PKCS12. Beispiel:

```
server.ssl.key-store-type=PKCS12
```

Schlüsselspeicher-Alias

Diese Property gibt den Alias an, der zur Identifizierung des Zertifikats und des privaten Schlüssels innerhalb des Schlüsselspeichers verwendet wird. Beispiel::

```
server.ssl.key-alias=noa-server
```



Anmerkung: Der Schlüsselspeicher muss ein gültiges SSL/TLS-Zertifikat und den dazugehörigen privaten Schlüssel enthalten.

Authentifizierung

The Natural OpenAPI server does not authenticate the client sending the request by itself, but rely on the authentication module of the Natural Subprogram Server. However, if the Natural Subprogram Server is configured to use an authentication method, you must define the appropriate configuration in the Natural OpenAPI Server. Defining the correct configuration will create the related definition in the OpenAPI document located at *http://localhost:OpenAPIServer_port/v3/api-docs*.

Der Natural OpenAPI Server authentifiziert den Client, der die Anfrage sendet, nicht selbst, sondern stützt sich auf das Authentifizierungsmodul des Natural Subprogram Server. Ist der Natural Subprogram Server jedoch für die Verwendung einer Authentifizierungsmethode konfiguriert, müssen Sie die entsprechende Konfiguration im Natural OpenAPI Server festlegen. Durch die Festlegung der korrekten Konfiguration wird die entsprechende Definition im OpenAPI-Dokument unter *http://localhost:OpenAPIServer_port/v3/api-docs* erstellt.

Es gibt drei verschiedene Möglichkeiten, die Authentifizierung zu definieren:

- [Verwendung der Basic-Authentifizierung](#)
- [Verwendung eines OAuth2-Tokens](#)
- [Verwendung anonymer Anfragen](#)

Verwendung der Basic-Authentifizierung

Wird die Property `com.softwareag.natural.openapi.web.config.authenticationType` auf `basic` gesetzt, muss die Client-Anwendung Benutzername und Passwort in Base64-Kodierung (definiert in RFC-7617) übermitteln.

Verwendung eines OAuth2-Tokens

Das Abrufen des Tokens obliegt der Client-Anwendung, während die Token-Validierung vom Authentifizierungsmodul des Natural Subprogram Server durchgeführt wird.

Wird die Property `com.softwareag.natural.openapi.web.config.authenticationType` auf `oauth2` gesetzt, muss die Client-Anwendung gemäß den Anforderungen des OpenAPI-Dokuments ein Bearer-Token senden.

Die Client-Anwendung muss neben dem bereitgestellten Token den speziellen HTTP-Header `X-Impersonate-User` einfügen, um den zu impersonierenden Benutzer anzugeben.

Die folgende Konfiguration kann beim Testen mit dem Swagger-UI-Webclient verwendet werden. Das in den Natural OpenAPI Server eingebettete Swagger-UI unterstützt derzeit nur den Client Credentials-Ablauf. Wenn die OAuth2-Authentifizierung aktiviert ist, können Sie die `tokenURL` und die erforderlichen Bereiche definieren. Diese Werte werden im OpenAPI-Dokument angegeben, aber vom Natural OpenAPI Server nicht zur Authentifizierung verwendet. Im Falle einer Impersonierung muss diese nach einer erfolgreichen Token-Überprüfung durchgeführt werden.

Verwendung anonymer Anfragen

Wenn in der Datei *application.properties* kein Authentifizierungstyp definiert ist, gelten die Anfragen standardmäßig als anonym und sollten nicht authentifiziert werden. Es ist auch möglich, die Anfrage unter Verwendung des speziellen HTTP-Header *X-Impersonate-User* als anderer Benutzer auszuführen.

V Übersicht für Entwickler

Dieses Dokument behandelt die folgenden Themen:

Installation des NaturalONE OpenAPI Plugin
Konfiguration des NaturalONE OpenAPI Plugin
Dienste generieren und bereitstellen
Besondere Hinweise

18

Installation des NaturalONE OpenAPI Plugin

Das NaturalONE OpenAPI-Plugin ist erforderlich, um Natural OpenAPI-Dienste zu erstellen und zu verwalten.

Das Plugin läuft in der NaturalONE IDE. Falls NaturalONE noch nicht installiert ist, installieren Sie es bei der Installation des NaturalONE OpenAPI-Plugin.

➤ **Um das NaturalONE OpenAPI Plugin zu installieren:**

- 1 Öffnen Sie den Software AG-Installer und navigieren Sie zum Installationsverzeichnis.
- 2 Erweitern Sie den Knoten **NaturalONE** und wählen Sie **OpenAPI Development**.

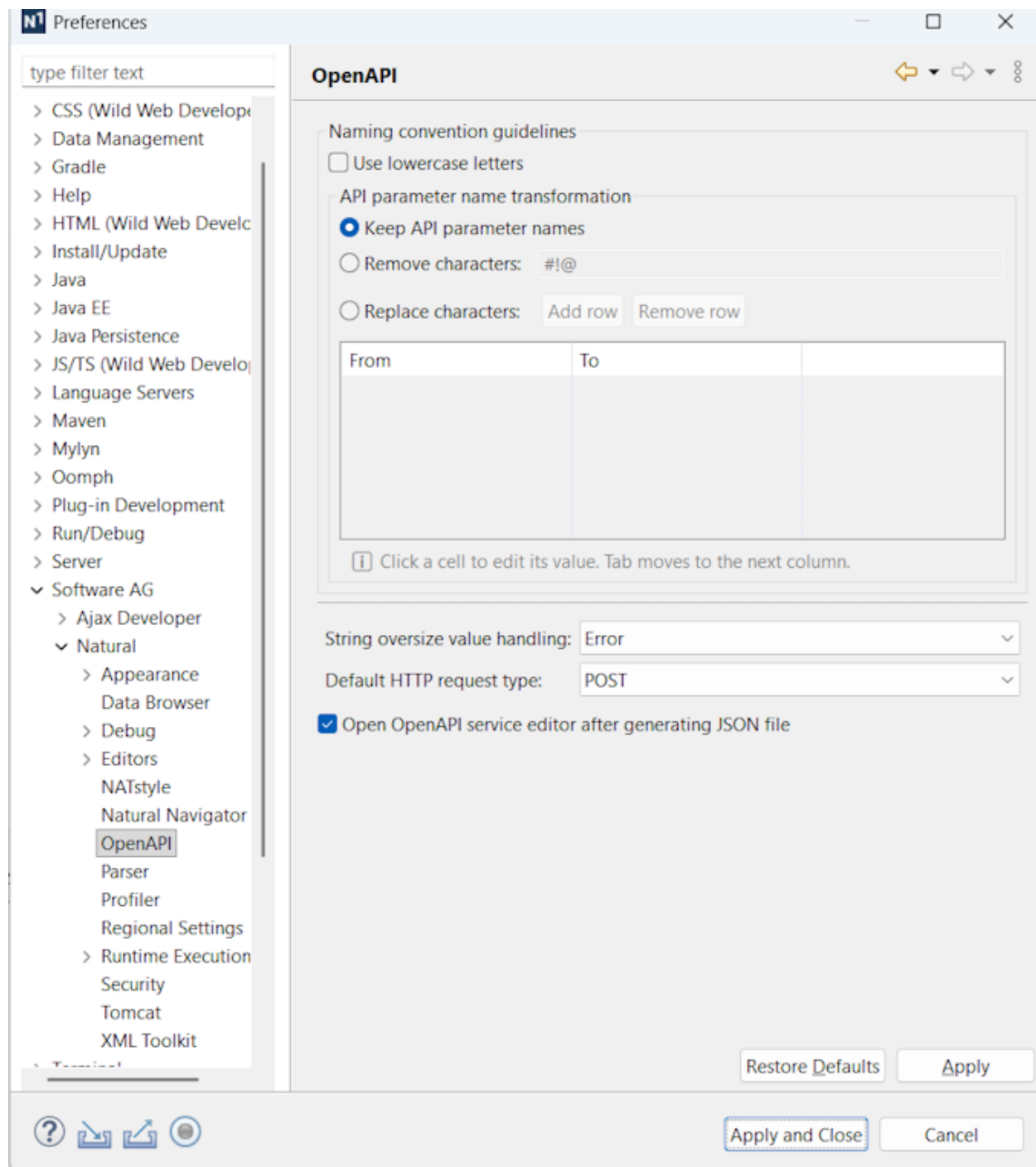
19

Konfiguration des NaturalONE OpenAPI Plugin

Die globalen Standardeinstellungen für die Generierung von OpenAPI-Diensten können im Eclipse-Fenster **Preferences** konfiguriert werden. Diese Einstellungen gelten für alle vom Plugin generierten Dienstdateien.

To access these settings, go to **Window** on the main menu and select **Preferences**. On the navigation tree, expand the **Software AG** node, **Natural** node, and then select **OpenAPI**.

Um auf diese Einstellungen zuzugreifen, wählen Sie im Hauptmenü **Window** und dann **Preferences**. Wählen Sie in dem Verzeichnis unter dem Knoten **Software AG** den Knoten **Natural** und wählen Sie anschließend **OpenAPI**.



Die Einstellungen im Fenster **Preferences** legen die Standardwerte fest, die bei der Generierung von OpenAPI-Diensten verwendet werden, wenn eine entsprechende Property im Subprogramm selbst nicht explizit definiert wurde.

Die folgenden Einstellungen sind verfügbar:

Use lowercase letters

Aktivieren Sie dieses Kontrollkästchen, um Parameternamen während der Generierung von OpenAPI-Diensten automatisch in Kleinbuchstaben umzuwandeln.

API parameter name transformation

Diese Einstellung bietet Optionen zur automatischen Umwandlung von OpenAPI-Parameternamen. Die folgenden Optionen sind verfügbar:

- **Keep API parameter names**

Bei Wahl dieser Option bleiben die Parameternamen unverändert.

- **Remove characters**

Bei Wahl dieser Option werden bestimmte Zeichen aus den Parameternamen entfernt. In einem zugehörigen Eingabefeld können Sie die zu entfernenden Zeichen angeben (z. B. #, ! oder @).

- **Replace characters**

Bei Wahl dieser Option können Sie Zuordnungen (Mappings) definieren, die festlegen, welche Zeichen oder Zeichenfolgen ersetzt werden sollen und welche Zeichen oder Zeichenfolgen anstelle der zu entfernenden Zeichen als Ersatz verwendet werden sollen.

String oversize value handling

Diese Einstellung legt das Standardverhalten bei der Behandlung überlanger Zeichenfolgenwerte fest. Diese Einstellung kann für einen einzelnen Dienst im OpenAPI-Editor überschrieben werden.

Die folgenden Optionen stehen zur Verfügung:

- **Truncate**

Diese Option kürzt den Wert, wenn er die zulässige Größe überschreitet.

- **Error**

Diese Option löst einen Fehler aus, wenn der Wert die zulässige Größe überschreitet.

Default HTTP request type

Diese Einstellung legt die Standard-HTTP-Anfragemethode fest, die beim Öffnen des OpenAPI-Editors ausgewählt wird, sofern im Subprogramm kein HTTP-Anfragetyp definiert wurde.

Open OpenAPI service editor after generating JSON file

OpenAPI-Service-Editor nach der Erstellung der JSON-Datei öffnen.

Diese Einstellung legt fest, ob der OpenAPI-Editor automatisch geöffnet wird, wenn die Funktion `Generate OpenAPI Service` ausgelöst wird.

Standardmäßig ist dieses Kontrollkästchen aktiviert. Ist es deaktiviert, wird die Service-Datei sofort erstellt, ohne dass der Editor geöffnet wird.

Wenn mehrere Subprogramme ausgewählt sind und die Funktion `Generate OpenAPI Service` ausgelöst wird, wird diese Einstellung umgangen und der OpenAPI-Editor nicht geöffnet.

20

Dienste generieren und bereitstellen

■ OpenAPI-Ressourcen generieren	92
■ OpenAPI-Ressourcen im Editor anzeigen	94

Dieses Dokument behandelt die folgenden Themen:

OpenAPI-Ressourcen generieren

OpenAPI-Dienste erstellen

Um OpenAPI-Dienste zu erstellen, muss der Natural OpenAPI Server die Definition eines Natural-Subprogramms einlesen. Diese Informationen werden in einer JSON-Datei für den Dienst modelliert und gespeichert.

Jede Dienst-JSON-Datei repräsentiert ein einzelnes Natural-Subprogramm, das die folgenden Informationen enthält:

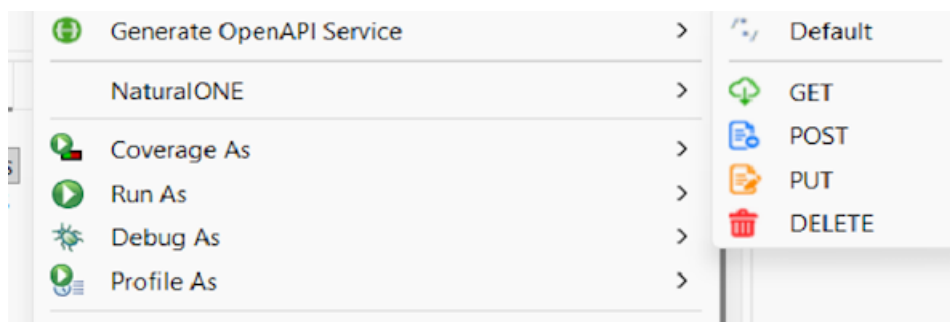
- Name des Natural-Subprogramms und der Natural-Bibliothek.
- OpenAPI-Schema-Mappings zwischen OpenAPI-Parametern und Natural-Eingabe- und -Ausgabeparametern.
- Abfrageparameter, die ebenfalls auf Natural-Parameter gemappt werden können.



Anmerkung: Abfrageparameter unterstützen derzeit keine Array-Mappings.

➤ Um einen OpenAPI-Dienst zu generieren:

- 1 Klicken Sie mit der rechten Maustaste auf das gewünschte Subprogramm.
- 2 Wählen Sie **Generate OpenAPI Service**. Es wird das folgende Menü mit diesen Optionen angezeigt:



- **Default**

Standardeinstellung – Verwendet die im Subprogramm definierte HTTP-Methode. Ist im Subprogramm keine HTTP-Methode definiert, wird der in den **Preferences** konfigurierte Wert verwendet.

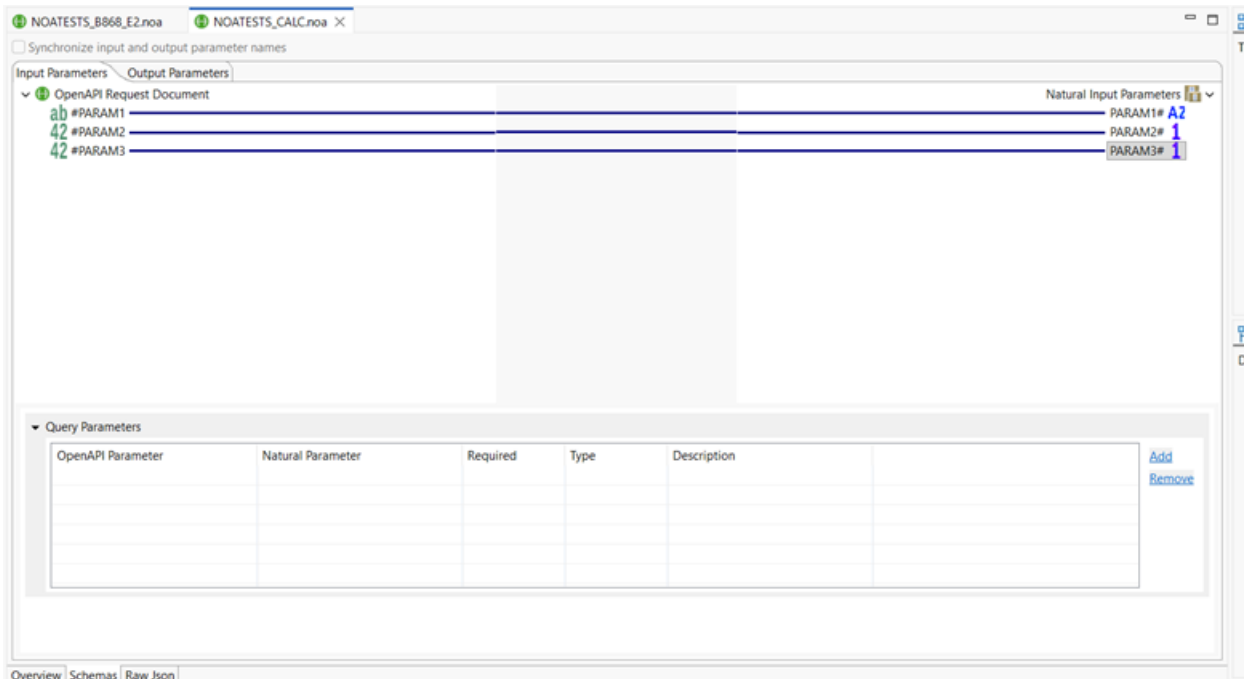
■ **GET / POST / PUT / DELETE**

Durch die Wahl einer bestimmten HTTP-Methode wird die Standardmethode überschrieben und die Methode festgelegt, die der Dienst verwenden soll.



Anmerkung: Array-Parameter unterstützen derzeit die HTTP-Methoden **GET** und **DELETE** nicht.

OpenAPI-Ressourcen im Editor anzeigen



Der OpenAPI Service Editor zeigt Informationen und Konfigurationseinstellungen der Dienste an. Er umfasst drei Registerkarten:

- [Registerkarte Overview](#)
- [Registerkarte Schemas](#)
- [Registerkarte Raw Json](#)

Registerkarte Overview

Die Registerkarte **Overview** zeigt allgemeine Informationen und Konfigurationseinstellungen für den Dienst an.

Auf dieser Registerkarte können Sie außerdem zusätzliche allgemeine Informationen wie Zusammenfassung, Beschreibung und Tags festlegen.

HTTP Request Type

HTTP-Anfragetyp: Gibt die vom Dienst verwendete HTTP-Methode an. Die unterstützten Methoden sind **GET**, **POST**, **PUT** und **DELETE**.

Behavior When a Wert Is Too Large

Verhalten bei zu großem Wert: Legt fest, wie der Dienst mit einem eingehenden OpenAPI-Wert umgeht, der die Größe des entsprechenden zugeordneten Natural-Parameters überschreitet.

Diese Einstellung kann für jeden einzelnen Dienst im Editor konfiguriert werden. Der Standardwert basiert auf der Option, die unter *String oversize value handling* (Behandlung übergroßer Zeichenfolgenwerte) im Eclipse-Fenster **Preferences** ausgewählt wurde.

Registerkarte Schemas

Diese Registerkarte enthält im unteren Bereich zwei weitere Registerkarten: **Input Parameters** und **Output Parameters**. Die Namen der Properties auf diesen beiden Registerkarten können synchronisiert werden, indem Sie das Kontrollkästchen **Synchronize input and output parameter names** aktivieren.

Registerkarte Input Parameters

Auf dieser Registerkarte wird das OpenAPI-Schema-Layout für die Natural-Eingabeparameter angezeigt. Das Layout bietet eine Zuordnungsschnittstelle (Mapping Interface) zwischen den Natural-Eingabeparametern und dem OpenAPI-Anfrage-Dokument.

Das OpenAPI-Schema-Layout besteht aus drei Bereichen und einem Abschnitt:

1. OpenAPI Request Document

Linker Baumbereich: Zeigt das OpenAPI-Dokument an. Ein Rechtsklick auf ein OpenAPI-Element im Baum zeigt die folgenden Optionen an:

- **Rename**
Umbenennen – Ändert den Namen des OpenAPI-Elements.
- **Restore Mapping**
Zuordnung wiederherstellen – Stellt eine zuvor entfernte Zuordnung wieder her.
- **Delete**
Löschen – Löscht den OpenAPI-Parameter, sofern er derzeit nicht zugeordnet ist.

2. Mapping Controls

Bereich Zuordnungssteuerung – Bietet die Möglichkeit, eine Zuordnung zu entfernen. Das Entfernen einer Zuordnung bedeutet, dass das OpenAPI-Element nicht mehr verwendet wird. Der entsprechende Natural-Parameter erhält seinen Standardwert (beispielsweise eine leere Zeichenkette für alphanumerische Parameter oder 0 für numerische Parameter).

3. Natural Input Parameters

Rechter Baumbereich – Zeigt die Natural-Parameter an. Ein Rechtsklick auf einen Natural-Parameter zeigt die folgenden Optionen an:

- **Map to Document**

Im Dokument zuordnen – Erstellt oder stellt eine Zuordnung zu einem Element des OpenAPI-Dokuments her.

- **Map to Query Parameter**

Zu Abfrageparameter zuordnen – Erstellt einen Abfrageparameter, der dem Natural-Parameter zugeordnet ist (nur aktiviert, wenn der Natural-Parameter nicht zugeordnet ist).

4. **Query Parameters**

Abschnitt Abfrageparameter: Ermöglicht das Hinzufügen und/oder Entfernen von Zuordnungen zwischen Abfrageparametern und Natural-Eingabeparametern. Abfrageparameter werden als Teil der URL übergeben und sind nicht im Hauptteil der REST-Anfrage enthalten.

Registerkarte Output Parameters

Auf dieser Registerkarte wird das OpenAPI-Schema-Layout für die Natural-Ausgabeparameter angezeigt. Das Layout bietet eine Zuordnungsschnittstelle (Mapping Interface) zwischen den Natural-Ausgabeparametern und dem OpenAPI-Antwortdokument.

Registerkarte Raw Json

Auf dieser Registerkarte wird die JSON-Datei des Dienstes angezeigt.

21

Besondere Hinweise

■ Entwicklung von Natural-Subprogrammen für Natsubsv	98
--	----

Dieses Kapitel behandelt Themen, die für Entwickler relevant sind.

Entwicklung von Natural-Subprogrammen für Natsubsv

Dieses Dokument behandelt die Unterschiede, die bestehen, wenn ein Natural-Subprogramm vom Natural Subprogram Server (Natsubsv) statt von einer Terminalsitzung ausgeführt wird.

Es handelt sich hierbei nicht um eine Einführung in die Natural-Programmierung – dazu siehe die Dokumentation des Basisprodukts Natural.

Die Unterschiede sind auf zwei Eigenschaften zurückzuführen:

- Kein Terminal. Es handelt sich um eine Serversitzung; es gibt weder einen Bildschirm noch eine Tastatur.
- Kein Zwischenzustand zwischen Aufrufen. Eine Sitzung bedient nacheinander die Anfragen vieler Endbenutzer und wird bei jeder Anfragegrenze zurückgesetzt.

Dieses Dokument behandelt die folgenden Themen:

- Ausführungsmodell
- Mapping von OpenAPI-Namen
- Nicht zulässige bzw. eingeschränkte Statements
- Zustandslosigkeit
- Transaktionen und Arbeitsdateien (Work Files)
- Fehlerbehandlung
- Systemvariablen
- Checkliste

Ausführungsmodell

- Das Subprogramm wird über einen `CALLNAT` aufgerufen und über sein normales Ende oder über `ESCAPE ROUTINE` verlassen.
- Der `DEFINE DATA PARAMETER`-Block ist der einzige Datenkanal zum und aus dem Subprogramm.
- Jede Anfrage benennt ihre eigene Bibliothek und ihren eigenen Benutzer; beide werden vom Server vor dem `CALLNAT` festlegt.
- Die Sitzung wird an der Anfragegrenze zurückgesetzt, sodass jeder Aufruf aus einem definierten Zustand startet und keine Spur hinterlässt.

Mapping von OpenAPI-Namen

In einem Subprogramm können Sie den Namen des API-Parameters definieren, der auf einen Natural-Parameter gemappt werden soll.

Dies geschieht durch Hinzufügen des Präfixes `oapi.n:`, gefolgt vom gewünschten Eigenschaftsnamen, im Kommentar des Feldes.

Beispiel in Natural:

```
#PARAM1 (A1) /* oapi.n:nameParm IN
```

Dadurch wird ein API-Parameter namens `nameParm` erstellt, der auf den Natural-Parameter `#PARAM1` gemappt wird.

Nicht zulässige bzw. eingeschränkte Statements

- **Nicht zulässige Statements**
- **Eingeschränkt verwendbare Statements**
- **Verursachen einer Sperrung nicht zulässig**

Nicht zulässige Statements

Statement	Grund
TERMINATE	Beendet die vom Server verwaltete Sitzung. Die Anfrage wird abgebrochen und die Ausgabeparameter werden nicht zurückgeschrieben.
STOP	Verlässt den CALLNAT, ohne die Ausgabeparameter zurückzuschreiben. Verwenden Sie ESCAPE ROUTINE.
FETCH	Startet ein Programm innerhalb der Anfrage. Der Rest des Subprogramms wird nicht ausgeführt und die Ausgabeparameter werden nicht zurückgeschrieben.
RUN	Erfordert Quellcode und einen Compiler in einer Sitzung, die Anfragen bedient.
LOGON	Die Bibliotheksumgebung wird vom Server pro Anfrage eingerichtet.
REINPUT	Erfordert ein Terminal.
Maps (INPUT USING MAP, WRITE USING MAP), PROCESS PAGE	Erfordert ein Terminal.
SET KEY	Erfordert ein Terminal. Die PF-Tasteneinstellungen werden pro Anfrage zurückgesetzt.
SET CONTROL	Erfordert ein Terminal.
OPEN CONVERSATION, CLOSE CONVERSATION	Konversationaler RPC wird nicht unterstützt.

Statement	Grund
DEFINE DATA CONTEXT	Kontextvariablen sind die Daten einer Konversation, und es gibt keine. Der Bereich gehört zu dem einzelnen CALLNAT, der ihn erstellt, und wird mit diesem freigegeben, sodass er nichts von einem Aufruf zum nächsten überträgt. Verwenden Sie den PARAMETER-Block.

Eingeschränkt verwendbare Statements

Statement	Was ist möglich?
STACK	Daten können auf den Stack gelegt und innerhalb derselben Anfrage wieder ausgelesen werden. Kommandos werden nicht ausgeführt.
INPUT	Nur ohne Terminal-E/A, d. h. beim Lesen von Daten vom Stack. Jedes INPUT-Statement, das einen Bildschirm anspricht, schlägt fehl.

Verursachen einer Sperrung nicht zulässig

Ein gesperrtes Subprogramm blockiert die Sitzung, in der es ausgeführt wird, und damit jede Anfrage, die von diesem Subprogramm bedient worden wäre.

- Warten Sie nicht auf benutzerseitige Interaktionen, externe Ereignisse oder an anderer Stelle vorhandene Sperren.
- Halten Sie CALLS zu 3GL-Modulen kurz. Der Server kann einen laufenden 3GL-Aufruf nicht unterbrechen, sodass die Sitzung so lange belegt bleibt, wie der Aufruf dauert.
- Halten Sie Arbeitseinheiten kurz. Datenbanksperrungen werden bis zu einem END TRANSACTION oder BACKOUT TRANSACTION aufrechterhalten, sodass eine lange Arbeitseinheit andere Anfragen von denselben Datensätzen fernhält.

Das Gleiche gilt für das Subprogramm als Ganzes: Während seiner Laufzeit steht die Sitzung für alle anderen nicht zur Verfügung.

Zustandslosigkeit

Jeder Aufruf muss für sich allein stehen: Er darf keine Daten weitergeben und darf nicht auf Daten zurückgreifen, die von einem früheren Aufruf hinterlassen wurden.

Folgende Elemente werden an jeder Anfragegrenze in einen definierten Zustand zurückversetzt:

- Globale Datenbereiche, einschließlich des globalen Datenbereichs mit Namen COMMON.
- Anwendungsunabhängige Variablen (DEFINE DATA INDEPENDENT).
- Session-Parameter, einschließlich derer, die mit SET GLOBALS und *LANGUAGE geändert wurden.
- Die Profilparameter zur Angabe von Maximalwerten MADIO, MAXCL, MAXERR und LT.
- Steplib-Ketten.
- Retained Sets und Adabas-Kommandokennungen (CIDs/Command IDs).

■ PF-Tasteneinstellungen.

Ein globaler Datenbereich (GDA) ist zu Beginn jeder Anfrage leer, sofern das Subprogramm ihn nicht selbst füllt, und ein durch `RETAINED` in einer Anfrage zurückbehaltener Set existiert in der nächsten Anfrage nicht mehr.

Auch mit Kontextvariablen (`DEFINE DATA CONTEXT`) lässt sich dies nicht umgehen. Ohne eine Konversation stellen sie keinen Datenbereich dar, der sich über Aufrufe hinweg erstreckt: Der Kontext existiert nur für den einen `CALLNAT` und ist nach dessen Rückkehr nicht mehr vorhanden, sodass er weder einen Zustand weitergeben noch von der nächsten Anfrage gelesen werden kann. Weitere Informationen finden Sie unter *Nicht zulässige Statements*.

*ETID und die darunter gespeicherten ET-Daten gehören zur Sitzung, nicht zum Endbenutzer. Sie werden von allen Anfragen gemeinsam genutzt, die die Sitzung bedient, und dürfen nicht zur Übertragung benutzerspezifischer Zustände verwendet werden.

Transaktionen und Arbeitsdateien (Work Files)

Beenden Sie Ihnen gehörende Arbeitseinheiten, d.h. vervollständigen Sie jede einzelne Arbeitseinheit mit `END TRANSACTION` oder `BACKOUT TRANSACTION`, bevor Sie sie zurückgeben. Der Server führt kein Festschreiben (Commit) durch.

Bei Erreichen der Anfragegrenze nimmt der Server normalerweise ein Rollback der Transaktion vor (`BACKOUT TRANSACTION`) und schließt offene Arbeitsdateien. Unter Linux können Sie beide Verhaltensweisen mit den Parametern `backoutTransactionPerRequest` und `closeWorkFilesPerRequest` deaktivieren; andernfalls werden beide Verhaltensweisen immer ausgeführt.



Anmerkungen:

1. Eine Arbeitseinheit, die nicht innerhalb der Anfrage festgeschrieben (committed) wird, ist verloren.
2. Datensätze dürfen nicht über Aufrufe hinweg gehalten werden.
3. Eine Arbeitsdatei muss innerhalb einer Anfrage geöffnet, verwendet und geschlossen werden. `CLOSE MODE=USER` hält sie nicht über die Anfragegrenze hinweg offen.

Fehlerbehandlung

Fehler, die während der Verarbeitung einer Anfrage auftreten, werden vom Server abgefangen und dem Aufrufer mit Fehlernummer, Zeile, Status und Objektname gemeldet.

Es wird empfohlen, das Statement `ON ERROR` zu verwenden, um Fehler als Daten zurückzugeben, auf die der Aufrufer reagieren kann – einen Fehlercode und eine Meldung in den Ausgabeparametern – und unbehandelte Fehler für wirklich unerwartete Bedingungen zu reservieren. Führen Sie die Bereinigung auf dem Fehlerpfad genau wie auf dem Erfolgspfad durch.

Systemvariablen

Variable	Wert
*SERVER-TYPE	CALLNAT
*USER	Der Benutzer der aktuellen Anfrage in Großbuchstaben, bis zu 8 Zeichen.
*NET-USER	Der Benutzer der aktuellen Anfrage, wie bei *USER. Unter z/OS enthält diese Variable stattdessen den Benutzer der vom Server gestarteten Sitzung – denselben Wert wie *INIT-USER – und identifiziert daher nicht den anfragenden Benutzer.
*LIBRARY-ID	Der Name der von der aktuellen Anfrage angegebene Bibliothek.
*INIT-USER, *INIT-ID, *ETID	Die vom Server gestartete Sitzung, nicht der anfragende Benutzer. Unter z/OS enthält *INIT-ID eine Sitzungsnummer, die der Server bei jeder neuen Sitzung erhöht.
*DEVICE	Ein Batch-Gerät.

Checkliste

- ☐ Kein `TERMINATE`, `STOP`, `FETCH`, `RUN` und `LOGON`.
- ☐ Kein `REINPUT`, keine Masken (Maps), kein `SET KEY`, kein `SET CONTROL`; `INPUT` nur ohne Terminal-Ein-/Ausgabe.
- ☐ Kein `OPEN CONVERSATION`, `/CLOSE CONVERSATION`, kein `DEFINE DATA CONTEXT`.
- ☐ Das Subprogramm las nur Werte, die es selbst gesetzt hatte; es las keine Werte aus einem globalen Datenbereich (GDA), aus unabhängigen Variablen oder aus einem früheren Aufruf beibehaltene Werte.
- ☐ Jede Arbeitseinheit endete auf jedem Pfad mit `END TRANSACTION` oder `BACKOUT TRANSACTION`.
- ☐ Jede geöffnete Arbeitsdatei wird auf jedem Pfad geschlossen.
- ☐ Nichts blockiert – weder das Subprogramm noch ein von ihm aufgerufenes 3GL-Modul.
- ☐ Der Endbenutzer wird über *USER, nicht über *NET-USER, *INIT-USER oder *ETID identifiziert.