ZABBIX '26 CONFERENCE GERMANY

Practical Use Cases for the New Nested Low-Level Discovery Feature in Zabbix

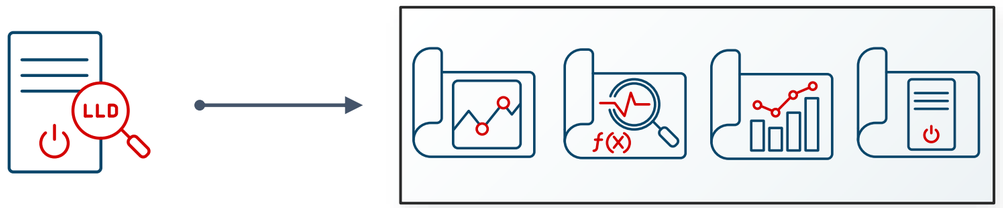
Stefan Matzek
Zabbix Trainer & Consultant
IntelliTrend GmbH

ZABBIX PREMIUM PARTNER

IntelliTrend IT-Services

Low-Level Discovery

LLD erkennt automatisch wiederkehrende Komponenten eines Geräts und erstellt daraus passende Entitäten aus Prototypen.



Kernfakten: Klassische LLD

- Seit Zabbix 2.0 verfügbar
- Regel liest JSON/Key-Values und füllt {#LLD}-Makros
- Erstellt automatisch Items, Triggers, Graphen und Hosts durch Prototypen
- Läuft periodisch und hält Objekte synchron
- Aufräumen über "Keep lost resources"

Was ist NLLD?

Nested Low-Level Discovery (NLLD) erzeugt keine Items direkt, sondern generiert weitere Discovery-Regeln.

Native Verschachtelung:

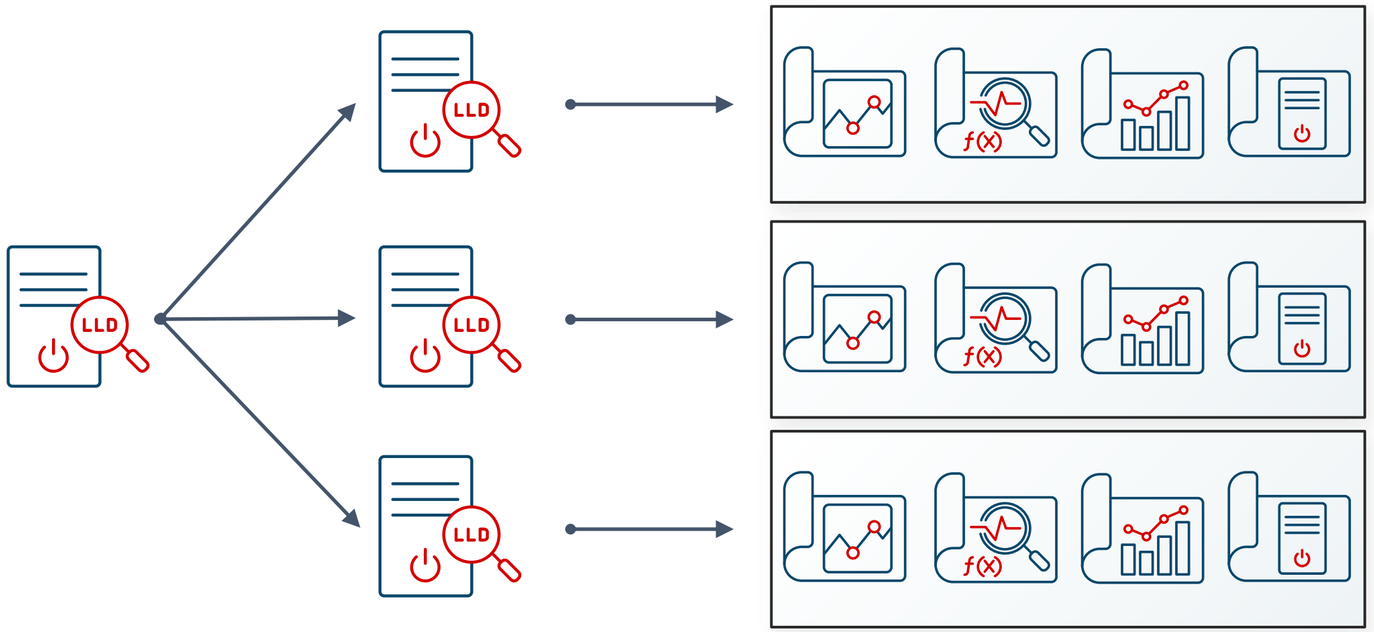
Selbst mehrstufige Hierarchien lassen sich direkt in Zabbix abbilden – ganz ohne externe Skripte oder Workarounds.

Kontext-Vererbung:

Makros der übergeordneten Discovery werden automatisch an alle Sub-Discovery-Regeln durchgereicht.

Klare Struktur:

Aufeinander aufbauende LLDs halten die Konfiguration übersichtlich.



Standard-LLD vs. Nested-LLD

Standard-LLD

- **Flach & Starr:** Nur eine Discovery-Ebene pro Regel möglich.
- **Daten-Overhead:** Oft gigantische, redundante JSON-Payloads nötig.
- **Isoliert:** Makros gelten nur in ihrer eigenen kleinen Bubble.
- **Flaschenhals:** Flache Kombinationslisten belasten Server-CPU und Datenbank.

Nested-LLD

- **Dynamisch & Tief:** Child-Regeln entstehen on-the-fly aus Parent-Funden.
- **Präzise:** Gezielte Filterung auf Teilbäumen (JSONPath) statt Massenabruf.
- **Vernetzt:** Parent-Makros stehen in Child-Regeln voll zur Verfügung.
- **Ressourcenschonend:** Weniger Netzwerk-Abrufe durch gemeinsame JSON-Bäume.

Der „Nested“ Type: Datenübergabe verstehen

Der Master liefert die Daten

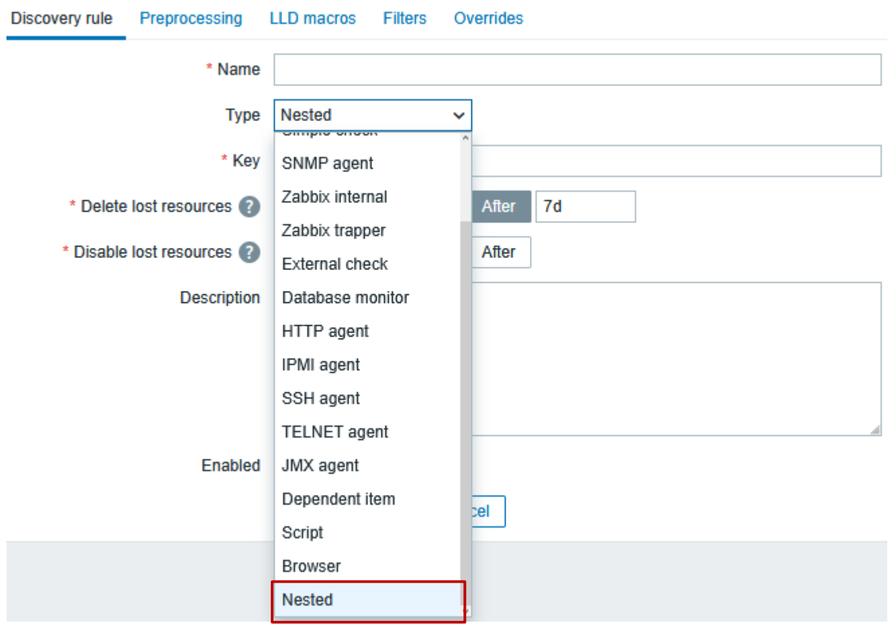
Eine NLLD-Kette beginnt fast nie mit dem Typ „Nested“. Die erste Regel ist meist ein HTTP-Agent oder ein anderes Item, das das große JSON-Paket holt.

Der Typ „Nested“ als „Konsument“

Nur die untergeordneten Discovery-Regeln (Child-Rules) könnten den neuen Typ „Nested“ erhalten.

LLD-Prototypen müssen nicht Type „Nested“ haben

Bei „Chatty APIs“ benötigt jede Ebene eigene Requests. Hier nutzt der Prototyp z. B. den Typ „HTTP-Agent“, erbt aber trotzdem alle Makros der Parent-Regel.

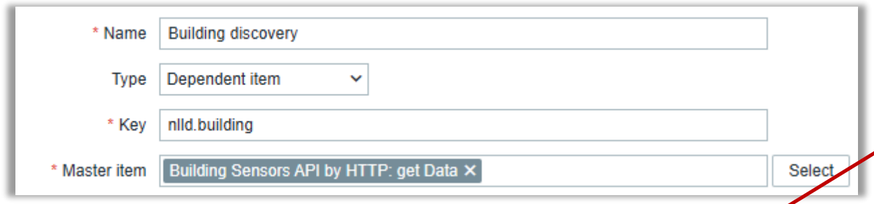


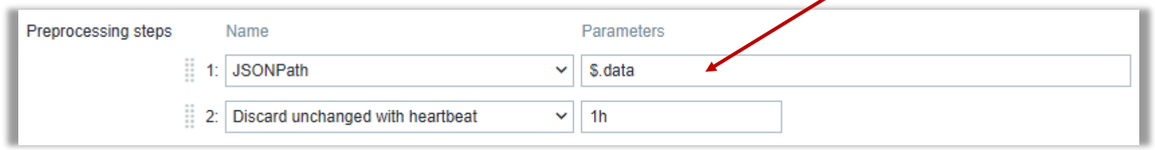
Nested Low-Level Discovery

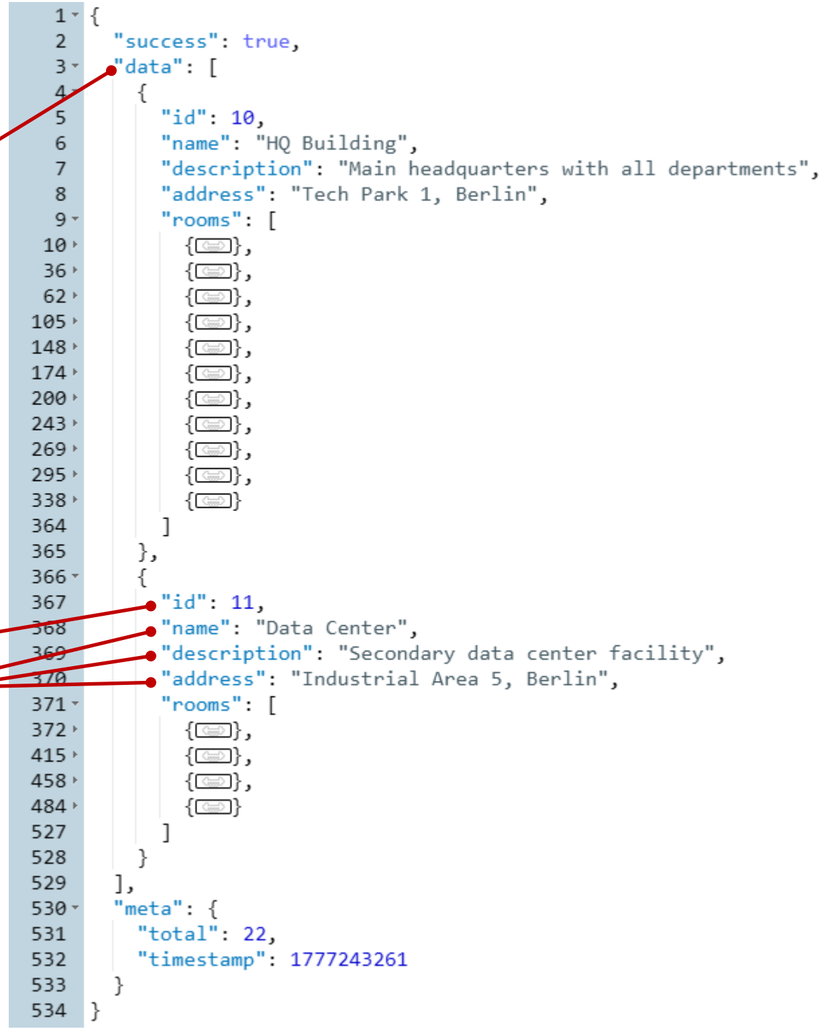
Beispiel - Chunky API

NLLD-Beispiel - Chunky API

Gebäude Discovery







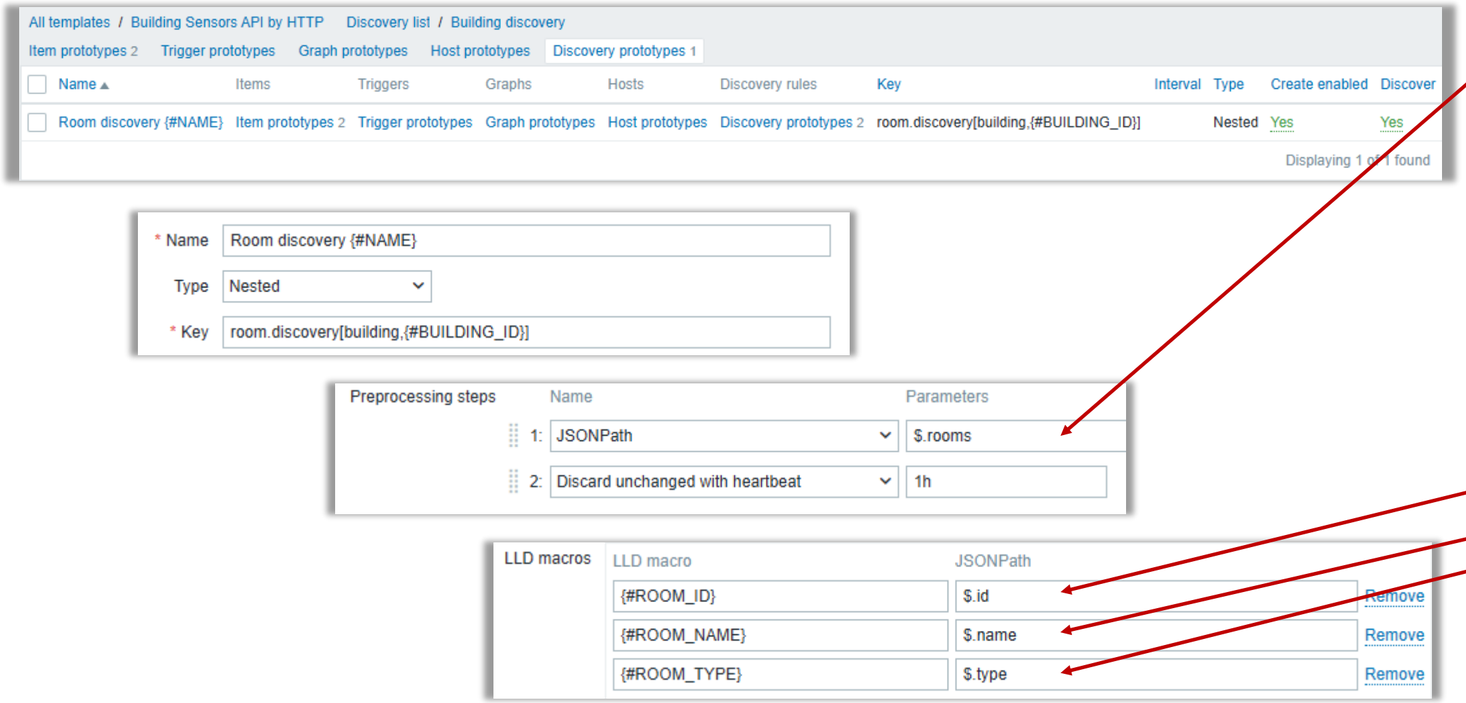
In einem Master-Item wird ein mehrschichtiges JSON gesammelt. Diese Ebene unterscheidet sich nicht von regulären LLD.

ZABBIX PREMIUM PARTNER

IntelliTrend IT-Services

NLLD-Beispiel - Chunky API

Raum Discovery




```
{
  "id": 11,
  "name": "Data Center",
  "description": "Secondary data center facility",
  "address": "Industrial Area 5, Berlin",
  "rooms": [
    {
      "id": 111,
      "building_id": 11,
      "name": "Server Hall A",
      "type": "serverroom",
      "floor": 0,
      "capacity": 6,
      "sensors": [...]
    },
    {
      "id": 112,
      "building_id": 11,
      "name": "Server Hall B",
      "type": "serverroom",
      "floor": 0,
      "capacity": 6,
      "sensors": [...]
    },
    {
      "id": 113,
      "building_id": 11,
      "name": "Network Room",
      "type": "serverroom",
      "floor": 0,
      "capacity": 3,
      "sensors": [...]
    },
    {
      "id": 114,
      "building_id": 11,
      "name": "UPS Room",
      "type": "storage",
      "floor": 0,
      "capacity": 2,
      "sensors": [...]
    }
  ]
}
```

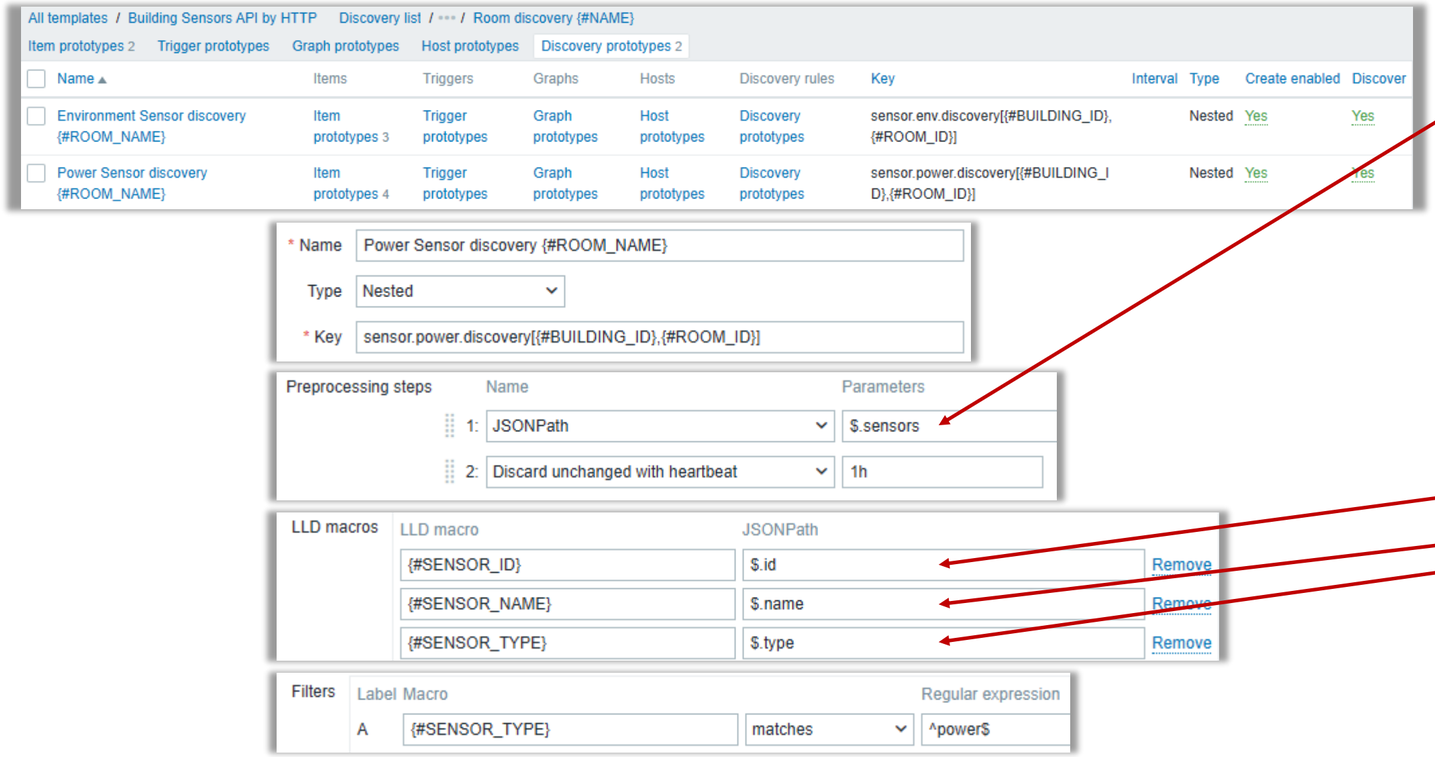

Das nächste Level erfordert eine Nested Low-Level Discovery
um alle Räume der Gebäude zu identifizieren.

ZABBIX PREMIUM PARTNER

IntelliTrend IT-Services

NLLD-Beispiel - Chunky API

Sensor Discovery

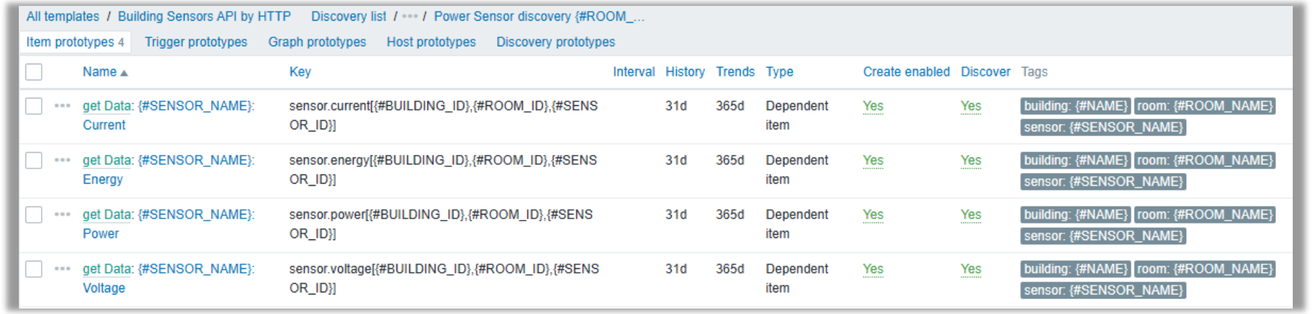


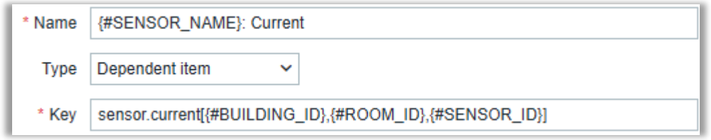

```
{
  "id": 111,
  "building_id": 11,
  "name": "Server Hall A",
  "type": "serverroom",
  "floor": 0,
  "capacity": 6,
  "sensors": [
    {
      "id": 1015,
      "room_id": 111,
      "name": "Environment Sensor - Rack A1",
      "type": "environment",
      "description": "Environmental monitoring for rack A1",
      "readings": [
        {
          "sensor_id": 1015,
          "temperature": 18.5,
          "humidity": 45.6,
          "co2": 666,
          "timestamp": 1777243261
        }
      ]
    },
    {
      "id": 1016,
      "room_id": 111,
      "name": "Smart Plug - PDU A",
      "type": "power",
      "description": "PDU power monitoring A",
      "readings": [
        {
          "sensor_id": 1016,
          "voltage": 230,
          "current": 0.105,
          "power": 24,
          "energy": 493752.28380446456,
          "timestamp": 1777243261
        }
      ]
    }
  ]
},
```


Das nächste Level erfordert eine Nested Low-Level Discovery
um alle Sensoren der Räume zu identifizieren.

NLLD-Beispiel - Chunky API

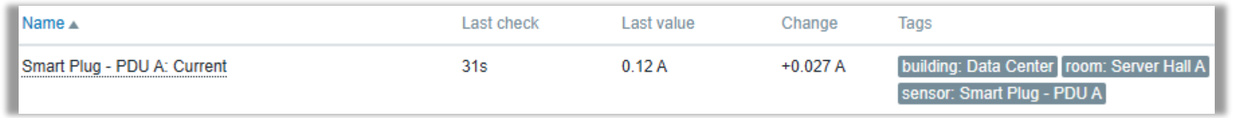
Metrik Extraktion





$.data[?(@.id =~ '^{#BUILDING_ID}$')].rooms[?(@.id =~ '^{#ROOM_ID}$')].sensors[?(@.id =~ '^{#SENSOR_ID}$')].readings[0].current.first()

$..sensors[?(@.id == '{#SENSOR_ID}')].readings[0].current.first()



Für das finale Level ist eine komplexe JSONPath-Logik erforderlich, wenn keine eindeutigen IDs pro Element vorhanden sind.


```
{
  "id": 111,
  "building_id": 11,
  "name": "Server Hall A",
  "type": "serverroom",
  "floor": 0,
  "capacity": 6,
  "sensors": [
    {
      "id": 1015,
      "room_id": 111,
      "name": "Environment Sensor - Rack A1",
      "type": "environment",
      "description": "Environmental monitoring for rack A1",
      "readings": [
        {
          "sensor_id": 1015,
          "temperature": 18.5,
          "humidity": 45.6,
          "co2": 666,
          "timestamp": 1777243261
        }
      ]
    },
    {
      "id": 1016,
      "room_id": 111,
      "name": "Smart Plug - PDU A",
      "type": "power",
      "description": "PDU power monitoring A",
      "readings": [
        {
          "sensor_id": 1016,
          "voltage": 230,
          "current": 0.105,
          "power": 24,
          "energy": 493752.28380446456,
          "timestamp": 1777243261
        }
      ]
    }
  ]
},
```


ZABBIX PREMIUM PARTNER

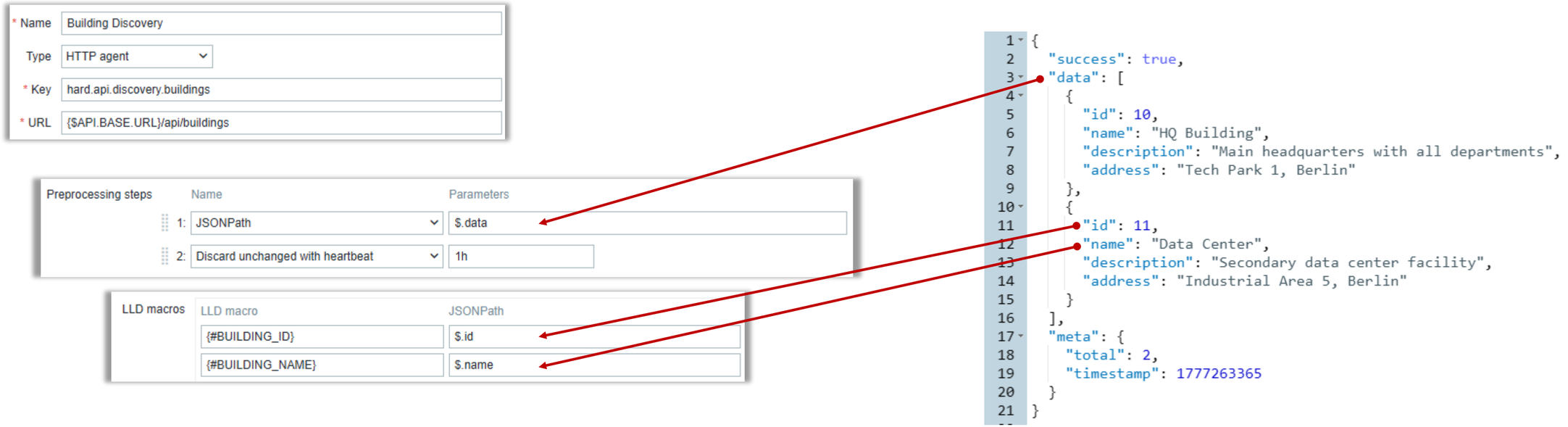
IntelliTrend IT-Services

Nested Low-Level Discovery

Beispiel - Chatty API

NLLD-Beispiel - Chatty API

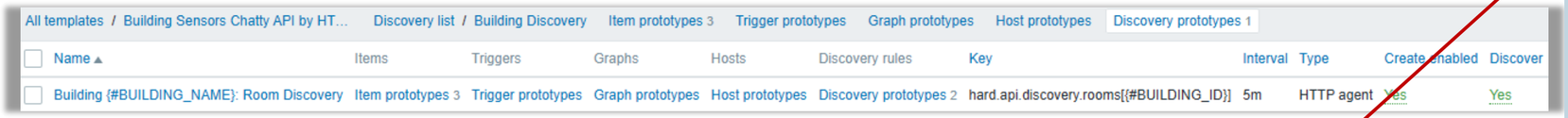
Gebäude Discovery

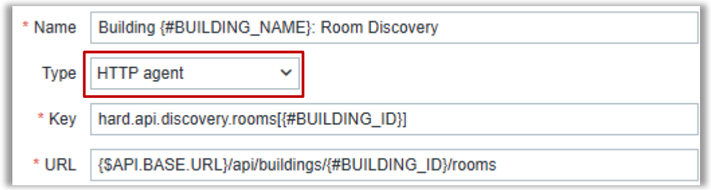


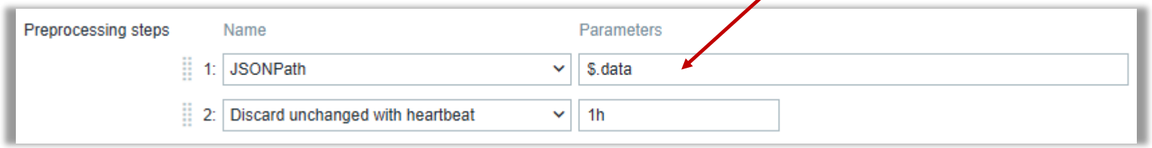
In Chatty APIs müssen alle einzelnen Ebenen individuell abgefragt werden (software-abhängig)

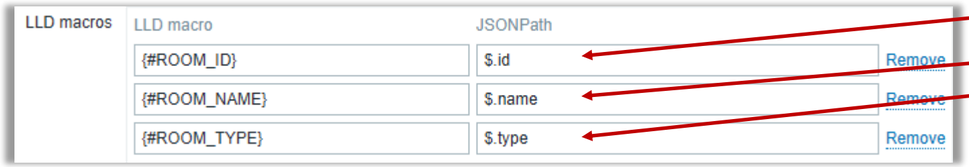
NLLD-Beispiel - Chatty API

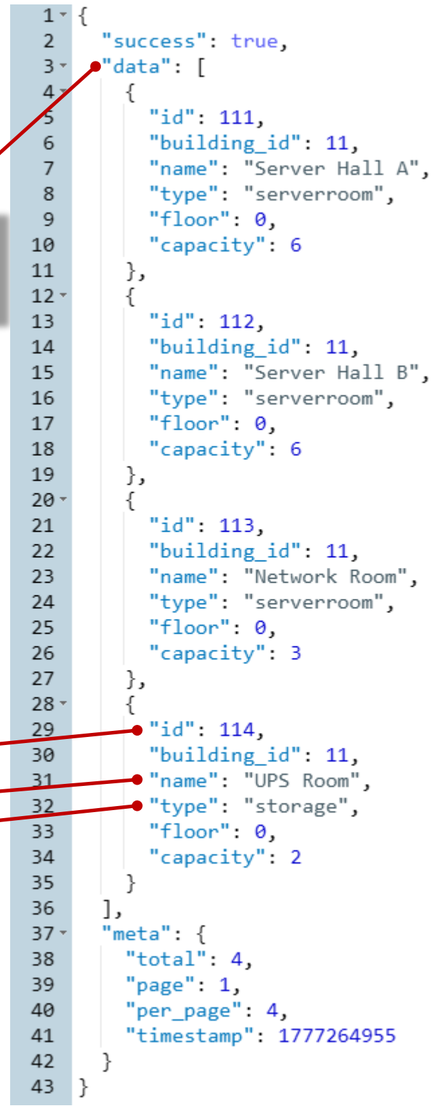
Raum Discovery











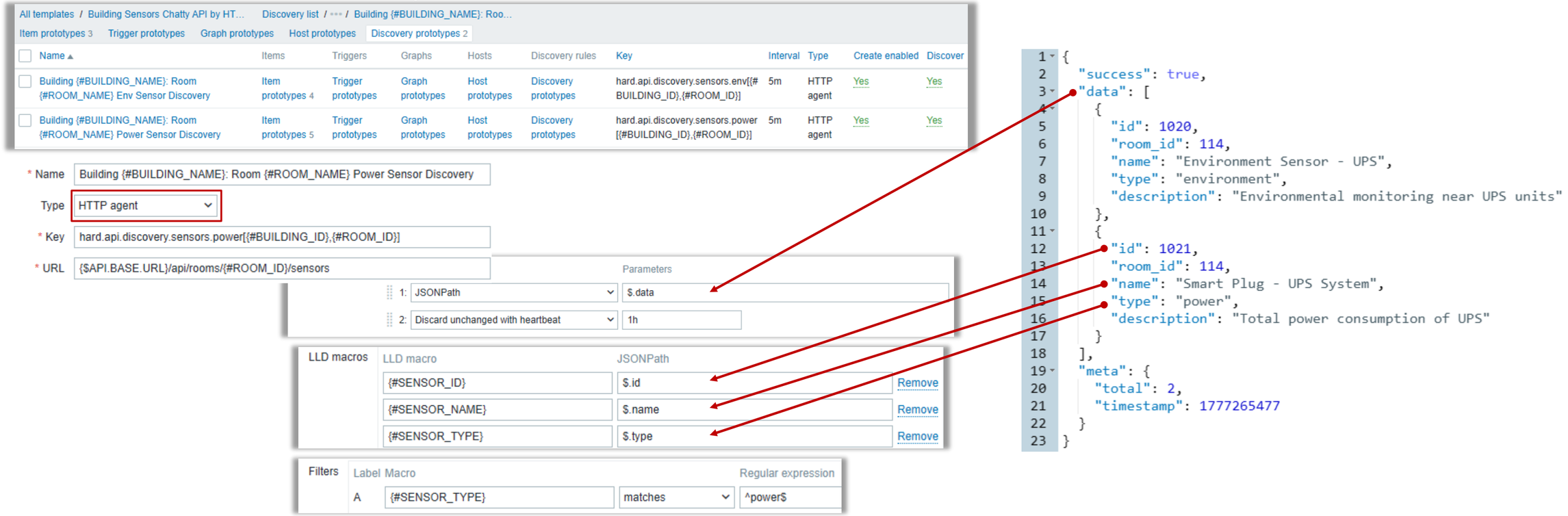
Nested Low-Level Discovery-Regeln können auch andere Typen als “Nested” haben. Die Raum Discovery bekommt Informationen von der Gebäude-Discovery, um tiefere Ebenen zu entdecken.

ZABBIX PREMIUM PARTNER

IntelliTrend IT-Services

NLLD-Beispiel - Chatty API

Sensor Discovery



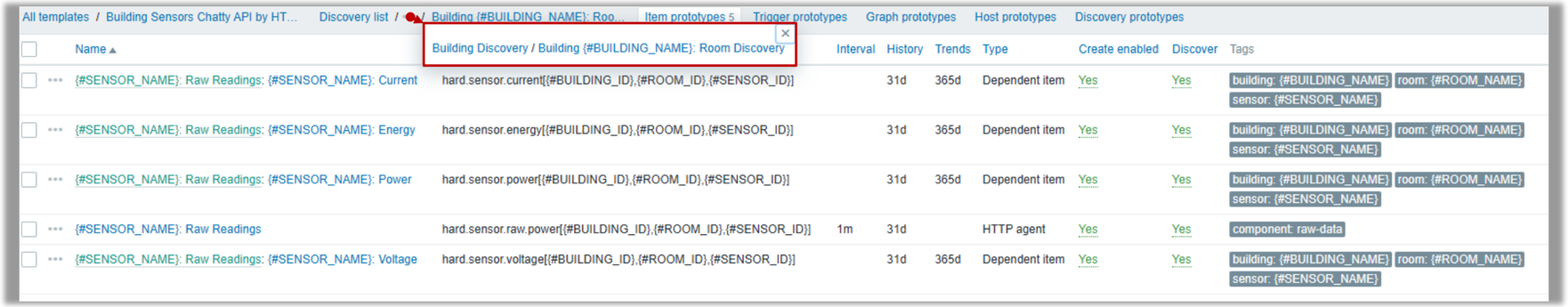
Die Sensor Discovery bekommt Informationen von der Raum- und Gebäude-Discovery, um tiefere Ebenen zu entdecken.

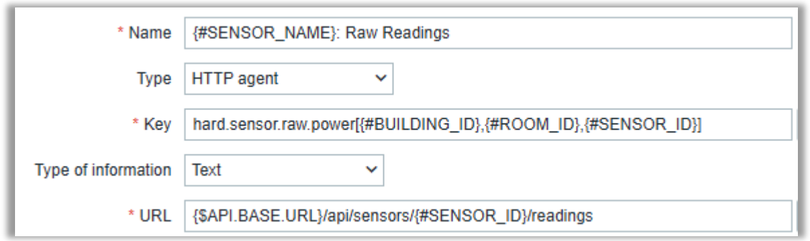
ZABBIX PREMIUM PARTNER

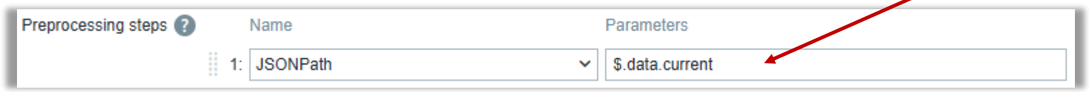
IntelliTrend IT-Services

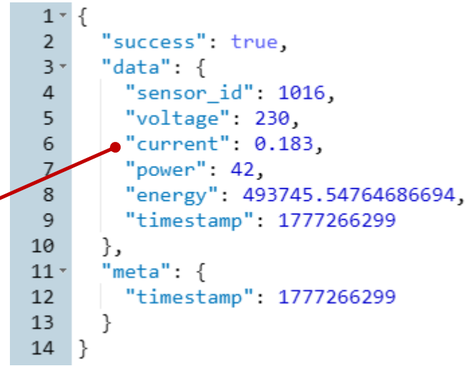
NLLD-Beispiel - Chatty API

Metrik Extraktion

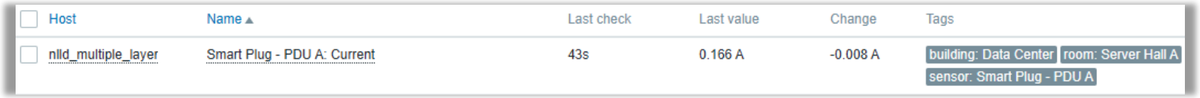








Finales Einsammeln der Metriken mittels HTTP-Agent und Dependent Items Prototypen auf einfachem Level.



Practical Use Cases

Redfish by HTTP

Aktuelle Redfish Templates

Name ▲	Hosts	Items	Triggers	Graphs	Dashboards	Discovery	Web	Vendor	Version	Linked templates	Linked to templates	Tags
DELL PowerEdge R660 by HTTP	Hosts	Items 14	Triggers 13	Graphs 4	Dashboards 1	Discovery 8	Web	Zabbix	7.4-2			class: hardware target: dell target: dell poweredge
DELL PowerEdge R720 by HTTP	Hosts	Items 14	Triggers 13	Graphs 4	Dashboards 1	Discovery 8	Web	Zabbix	7.4-3			class: hardware target: dell target: dell poweredge
DELL PowerEdge R740 by HTTP	Hosts	Items 14	Triggers 13	Graphs 4	Dashboards 1	Discovery 8	Web	Zabbix	7.4-3			class: hardware target: dell target: dell poweredge
DELL PowerEdge R750 by HTTP	Hosts	Items 14	Triggers 13	Graphs 4	Dashboards 1	Discovery 8	Web	Zabbix	7.4-2			class: hardware target: dell target: dell poweredge
DELL PowerEdge R820 by HTTP	Hosts	Items 14	Triggers 13	Graphs 4	Dashboards 1	Discovery 8	Web	Zabbix	7.4-3			class: hardware target: dell target: dell poweredge
DELL PowerEdge R840 by HTTP	Hosts	Items 14	Triggers 13	Graphs 4	Dashboards 1	Discovery 8	Web	Zabbix	7.4-3			class: hardware target: dell target: dell poweredge
HPE iLO by HTTP	Hosts	Items 2	Triggers 1	Graphs	Dashboards 1	Discovery 9	Web	Zabbix	7.4-0			class: hardware target: hpe target: ilo

Displaying 7 of 7 found

Aktuelle Zabbix 7.4 Templates für Redfish nutzen keine NLLD-Features.
Viele Templates für die verschiedenen Modelle mit leicht abweichenden Konfigurationen.

Practical Use Cases

Redfish by HTTP

- Dell PowerEdge R330
- Dell PowerEdge R520
- Dell PowerEdge R540
- Dell PowerEdge R730
- Dell PowerEdge R749
- Dell PowerEdge R750xs
- Dell PowerEdge R760
- Dell PowerEdge R770
- Dell PowerEdge T440
- Fujitsu D3384
- HPE ProLiant DL20Gen10Plus
- HPE ProLiant DL360Gen10
- Lenovo ThinkSystem SR650V2
- Supermicro H13SSW
- Supermicro X11DPH-T
- Thomas Krenn AMD single-CPU TA1508-CHEP tower server
- Thomas Krenn Intel single-CPU TI108 tower server

Name ▲	Triggers	Key	Interval	History	Trends	Type
Redfish chassis		redfish.chassis	5m	0		Script
Redfish chassis: Redfish chassis errors		redfish.chassis.errors		31d		Dependent item
Redfish managers: Redfish manager errors		redfish.managers.errors		31d		Dependent item
Redfish managers		redfish.managers	5m	0		Script
Redfish systems		redfish.systems	5m	0		Script
Redfish systems: Redfish systems errors		redfish.systems.errors		31d		Dependent item

Template	Name ▲	Items	Triggers	Graphs	Hosts	Discovery rules	Key	Interval	Type
Redfish by HTTP	Redfish chassis: Chassis	Item prototypes	Trigger prototypes	Graph prototypes	Host prototypes	Discovery prototypes 4	redfish.chassis.discovery		Dependent item
Redfish by HTTP	Redfish managers: Managers	Item prototypes	Trigger prototypes	Graph prototypes	Host prototypes	Discovery prototypes 1	redfish.managers.discovery		Dependent item
Redfish by HTTP	Redfish systems: Systems	Item prototypes 5	Trigger prototypes	Graph prototypes	Host prototypes	Discovery prototypes 5	redfish.systems.discovery		Dependent item

Redfish Template mit NLLD-Feature -
Kompatibel mit diversen Geräten

Work in Progress

ZABBIX PREMIUM PARTNER

IntelliTrend IT-Services

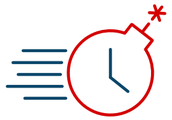
Einschränkungen mit Nested LLD

Komplexe Fehlersuche & Abhängigkeiten

Das Debuggen von Makros, die über mehrere Ebenen vererbt werden, ist unübersichtlich.

Auch das Bauen von dynamischen Trigger-Abhängigkeiten zwischen "Geschwister-Regeln" ist weiterhin schwierig.

Exponentielles Performance-Risiko



Mit jeder neuen Ebene multipliziert sich die Anzahl der generierten Regeln, Items und Trigger.

Ohne sehr strikte LLD-Filter kann der Zabbix-Server und die Datenbank schnell in die Knie gezwungen werden.

Steile Lernkurve & Setup-Aufwand

Das Konzept ist "hard to master".
Um JSONPath, Preprocessing und Makro-Vererbung fehlerfrei über mehrere Ebenen aufzubauen, ist massiver Konfigurationsaufwand und Zabbix-Expertenwissen nötig.

Zusammenfassung: NLLD in der Praxis

Gamechanger für komplexe Datenstrukturen

NLLD bricht die starre Ein-Ebenen-Grenze des klassischen LLD auf und ermöglicht native, mehrstufige Hierarchien ohne externe Skripte.

Effizienz durch Vererbung

Die automatische Weitergabe von Makros (Kontext-Vererbung) über alle Ebenen hinweg eliminiert redundante Datenabfragen und komplexe Workarounds.

Flexibilität für jede API

Egal ob ein massives JSON-Paket („Chunky“) oder viele gezielte Einzelabfragen („Chatty“) – NLLD lässt sich optimal an die Architektur der Zielsoftware

Planung ist alles

Der hohe Konfigurationsaufwand zahlt sich durch eine saubere, wartbare Struktur und deutlich reduzierte Last auf Zabbix-Server und Datenbank aus, sofern man die Skalierung im Blick behält.

ZABBIX '26
CONFERENCE
GERMANY

Vielen Dank für Ihre Aufmerksamkeit!

Stefan Matzek
Zabbix Trainer & Consultant
IntelliTrend GmbH

ZABBIX
PREMIUM PARTNER

IntelliTrend
IT-Services