# **XTrillion Core Bond Calculation Engine**

# **API Specification**

**Date: 24 July 2025**

## **1. Overview**

Xtrillion's bond analytics engine delivers institutional-grade calculations with a focus on clarity, accuracy, and Bloomberg compatibility. Our philosophy is to offer the essential, high-value metrics used in trading, portfolio management, and risk systems—without the burden of low-utility complexity.

This document outlines the technical foundation, calculation suite, authentication requirements, and design principles of our API, which provides programmatic access to Xtrillion Core.

### **1.1 What is Xtrillion Core?**

Xtrillion Core is the calculation engine and smart bond parser that powers all Xtrillion software. It is designed to be the "brain" of modern fixed income infrastructure.

**Product Structure:**

* **Xtrillion Core**: The standalone API product for integration into client systems.
* **Xtrillion Platform**: A full-stack fund management solution built on Xtrillion Core.

Licensing Xtrillion Core gives clients access to the validated engine behind the entire Xtrillion ecosystem.

## **2. Technical Foundation**

### **2.1 QuantLib-Based Professional Engine**

* Built on QuantLib, the industry-standard fixed income library.
* Extensively validated to match Bloomberg outputs for US Treasuries and other instruments.
* Solves for day counts, calendars, and business day conventions across markets.

### **2.2 Smart Bond Parser**

* **ISIN Recognition**: Auto-identifies bonds and fetches conventions from our database.
* **Description Parsing**: Accurately extracts bond features from text-based input.
* **Convention Mapping**: Maps to correct QuantLib parameters without manual intervention.

#### **2.2.1 Description Parsing Examples**

The Smart Bond Parser can identify bonds from various text formats without requiring an ISIN:

**Treasury Shorthand Example:**

A white rectangular object with a black border

AI-generated content may be incorrect.

**Parsed Result:**

A screenshot of a computer code

AI-generated content may be incorrect.

**Additional Supported Formats:**

* "UST 3% 08/15/52" - US Treasury with explicit formatting US date format
* "TREASURY 3 15/08/2052" - Treasury with year-only maturity
* "US TREASURY N/B, 3%, 15-Aug-2052" - Full Bloomberg-style description
* "T 3 15-Aug-52" - Ticker based notation

## **3. Core Calculation Suite**

The engine provides three tiers of analytics. The API field names follow a consistent naming convention explained in Section 3.1.

### **Tier 1: Essential Metrics (Bloomberg Validated)**

**Calculation API Field Purpose Bloomberg Match**

A screenshot of a phone

AI-generated content may be incorrect.

### **Tier 2: Risk & Option-Adjusted Analytics**

**Calculation API Field Purpose Bloomberg Match**

### A screenshot of a computer AI-generated content may be incorrect.

### **Tier 3: Spread Analytics (Curve Dependent)**

**Calculation API Field Purpose Bloomberg Match**

### **3.1 Compounding Basis and API Naming Convention**

Many fixed income metrics, particularly measures of yield and risk, vary depending on the yield's compounding frequency (the basis). For example, a bond with a semi-annual yield of 4.90% has an equivalent annual yield of 4.958%. To ensure clarity and prevent aggregation errors, Xtrillion follows a consistent naming pattern.

**Our Convention:**

* **Default/Base Field**: The API field name without a suffix (e.g., ytm, mod\_dur) returns the value calculated using the bond's **native payment frequency**. For most US and UK bonds, this is a semi-annual basis. This aligns with the default display on Bloomberg terminals.
* **Explicit Basis Suffixes**: To get a specific basis, use the \_[basis] suffix.
  + \_annual: Returns the value on an annually-compounded basis. This is crucial for portfolio-level aggregation.
  + \_semi: Explicitly requests the semi-annual basis.

**Affected Metrics:** The following metrics are dependent on the compounding basis and are available with \_annual and \_semi suffixes upon request (e.g., in the portfolio context):

|  |  |
| --- | --- |
| * ytm (Yield to Maturity) | * convexity |
| * ytw (Yield to Worst) | * pvbp |
| * mod\_dur (Modified Duration) | * oad (Option-Adjusted Duration) |
| * mac\_dur (Macaulay Duration) | * oas (Option-Adjusted Spread) |

This convention is demonstrated in the context=portfolio request in Section 5, which returns results for both semi\_annual\_basis and annual\_basis.

## **4. API Design: Context-Based Responses**

The API adjusts output based on context parameters, mimicking Bloomberg pages.

### **Default (no context)**

### A close-up of a text AI-generated content may be incorrect.

### **Context: DES (Descriptive Data)**

### A screenshot of a computer AI-generated content may be incorrect.

### **Context: YAS (Yield and Spread)**

## A cartoon of a dog AI-generated content may be incorrect.

## **5. API Request Examples**

### **5.1 Description-Based Bond Calculation**

**Request using Treasury shorthand:**

A screenshot of a computer code

AI-generated content may be incorrect.

**Response:**

### A screenshot of a computer code AI-generated content may be incorrect.

### **5.2 Portfolio Context**

**Note:** For portfolio-level cash flow calculations, the nominal (face amount) of each bond must be explicitly provided. If it is omitted, the API will not return portfolio cash flow data.

**Request:**

**A screenshot of a computer code

AI-generated content may be incorrect.**

**Response:**

A screenshot of a computer

AI-generated content may be incorrect.

**Request:**

## A screenshot of a computer code AI-generated content may be incorrect.

**Response:**

## **A number of numbers and symbols AI-generated content may be incorrect.**

## **6. Authentication & Access Control**

All API endpoints support secure access using API keys. Each client is assigned a unique key, and all activity is logged for auditing purposes. Keys can be provisioned for different environments including institutional, development, and demo access.

**Authentication Header:**

X-API-Key: [your\_api\_key]

Please contact us to request access credentials appropriate to your integration environment.

## **7. Smart Bond Parser Coverage**

The smart bond parser covers over 80% of use cases without requiring a static data feed. This eliminates the need for costly subscriptions for most instruments.

## **8. Quality Assurance & Precision**

* Test suite spans global bonds with edge cases.
* Bloomberg-precision validated.
* Accrued interest and other fields returned with 16-decimal precision.
* Error < 0.01 on a 1 billion trade.

## **9. Competitive Edge**

* **Proven Accuracy**: Matches Bloomberg outputs.
* **Context Awareness**: Smart responses tailored to use case.
* **Cost Reduction**: Cuts static data requirements by 80%.
* **Developer Friendly**: JSON-first design with intuitive fallback behaviour.
* **Description Parsing**: Calculate without ISIN or expensive data feeds.

## **10. Multi-Currency Support**

XTrillion Core supports multi-currency portfolios; however, yields can be client-driven regarding hedged versus unhedged calculations. Therefore, the system is optimally suited to US dollar-based portfolios.

## **11. Out of Scope Analytics**

To maintain our focus on providing a robust, self-sufficient engine and reduce dependency on complex external data feeds, the following analytics are currently considered out of scope:

* **Key Rate Durations (KRDs):** Calculating KRDs requires a complete, real-time benchmark yield curve. While our spread analytics use a benchmark curve, providing granular key rate risk is a more specialised function that introduces significant data dependency.
* **Asset Swap Spread (ASW):** ASW calculation is highly dependent on the relevant interest rate swap curve (e.g., SOFR, EURIBOR), which varies by currency and market convention. Supporting this would conflict with our goal of minimising reliance on external data feeds.

This approach ensures that Xtrillion Core delivers on its promise of providing essential, high-value metrics with maximum reliability and minimal external dependencies.

## **12. Future Roadmap**

Xtrillion Core is designed for extensibility. A planned enhancement is the integration with modern, AI-native frameworks. This will enable dynamic, intent-driven calculations and allow seamless interaction with LLM-powered interfaces and financial operating systems, reinforcing the long-term strategic value of the engine.

*Note: api.xtrillion.com is a placeholder for the production endpoint.*