ICS 35.240.40
CCS A 11

T

团　　体　　标　　准

T/ZFIDA 0008—2025

大模型金融领域可信应用参考框架

Reference Framework for Trustworthy Application of Large-scale Models in Finance

2025-12-04 发布　　2025-12-04 实施

中关村金融科技产业发展联盟　发布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则 第1部分：标准化文件的结构和起草规则》的规定起草。

本文件由蚂蚁科技集团股份有限公司提出。

本文件由中关村金融科技产业发展联盟归口。

本文件起草单位：蚂蚁科技集团股份有限公司、国投证券股份有限公司、北京知其安科技有限公司、中关村金融科技产业发展联盟、北京前沿金融监管科技研究院、浙江省网络空间安全协会、网联清算有限公司、青岛银行股份有限公司、公安部第三研究所（公安部网络安全等级保护评估中心）、上海第五空间信息科技研究院、兴业银行股份有限公司、华福证券有限责任公司、深圳市前海金融同业公会、中国邮政储蓄银行股份有限公司、中国光大银行股份有限公司、中国民生银行股份有限公司、中国平安银行股份有限公司、浙江网商银行股份有限公司、东方证券股份有限公司、世纪证券股份有限公司、工银科技有限公司、支付宝支付科技有限公司、北京赛博英杰科技有限公司、浙江蚂蚁密算科技有限公司、中关村互联网金融研究院、《信息安全研究》杂志社有限公司。

本文件主要起草人：韦韬、王宇、孙曦、刘焱、仲震宇、彭晋、陆碧波、刘书航、蓝潞、李维春、聂君、刘勇、张克、郭威、姜楠、曹亭亭、赵浚雅、方蕊、张园超、杜越、陈佃晓、王辽松、李变、邱康康、王勇、齐婕、朱易翔、何平、张笑东、王洪娉、李宁、谢琪、王钟菱、雷振锋、邓志为、赵汉杰、史佳涵、韩笑、丁焱、张爱雪、尹木涵、陆黎川、宋克亚、牛博强、何琰、贾凯、张庆、王舜、李剑峰、李剑铭、林永峰、姚刚、孙科伟、吴飞飞、宿悦茨、林志伟、谭晓生、孟繁强、高佩明、王磊、刘前伟、李泉、杨小强、齐柏尧、潘静、程斌、陈达、薛见新、胡钺琳。

引　言

以大模型为代表的新一代人工智能技术正加速迭代，为金融行业的智能化升级注入强劲动力 。目前，大模型已在一些任务简单、容错率高的场景（如智能客服、文案生成）中得到一定应用，并逐渐向金融投资、风险管理等任务复杂度高、专业性强、容错率低的核心领域渗透 。然而，要在此类高风险、高价值的场景中实现规模化落地，还需要进一步解决大模型在金融领域应用面临的风险与挑战，包括：首先是模型固有的“幻觉”现象，即生成与事实不符或毫无根据的内容，可能导致错误的投资建议或不合规的报告，造成直接的经济与法律后果；其次，金融领域的任务往往具有高度的复杂性，若不经过有效的任务拆分，其处理难度会超出当前大模型的能力上限，导致结果不可靠；同时，大模型对用户指令的遵循存在不确定性，尤其在处理复杂的多步骤推理任务时，可能出现指令误解或遗漏，进一步影响输出的可控性；此外，数据安全与隐私泄露、决策过程缺乏透明度等问题，也为金融机构带来一定的合规与运营挑战。

实践证明，仅仅致力于提升大模型自身的能力，难以从根本上消除这些内生的不确定性。因此，本文件倡导的理念是：不再追求大模型本身的完美，而是借鉴人类社会在管理复杂专业系统时采用的成功机制，如标准作业流程和检查清单（Checklist）等，通过系统工程的思维，为大模型构建一个外部的、确定性的、可验证的控制与保障体系。即使大模型在执行过程中出现偏差，该体系也能确保最终结果的正确性与可靠性。同时，这套体系也必须要经过真实金融业务场景的充分验证后，才能够在已验证场景下进行规模化应用。

这个控制与保障体系，在本文件中定义为大模型在金融领域的可信应用参考框架。需进一步阐述的是，人工智能的“可信（Trustworthiness）”内涵非常丰富，不仅涵盖了技术的准确性、安全性、鲁棒性和可解释性，还需要考虑伦理问题如公平性和隐私保护，以及社会影响如公众接受度和法律合规性等多个方面。在本文件所介绍的“可信应用参考框架”中，会更为关注大模型应用的专业性、可控性、真实性和安全性等属性。具体而言：

—专业性：系统的输出，无论是分析报告、代码还是客户对话，都需要在术语使用、逻辑推理和业务流程上符合金融领域的专业标准和最佳实践；

—可控性：系统的行为需要是可预测、可治理且可审计的。其核心业务逻辑和决策流程可由外部的、确定性的规则所主导，而非完全依赖模型的内部隐式推理；

—真实性：系统生成的所有事实性内容需要是准确的，并且能够被追溯到可信的数据源。需要建立主动的机制来识别和抑制模型幻觉；

—安全性：系统需要具备抵御外部攻击（如提示注入）和内部威胁的能力，确保数据在全生命周期内的机密性、完整性和可用性，并符合金融行业的安全法规。

通过这一框架，期望能够帮助行业中的开发者和应用者更好地推动大模型在金融领域中的应用深化和场景拓展。

大模型金融领域可信应用参考框架

1 范围

本文件提出了大模型在金融领域可信应用的参考框架和核心模块构成，并给出了技术实现指南。

本文件适用于金融领域的大模型应用系统的规划、设计、开发、部署和测试等。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 45288.1-2025 人工智能 大模型 第1部分：通用要求

GB/T 45438-2025 网络安全技术 人工智能生成合成内容标识方法

GB/T 45654-2025 网络安全技术 生成式人工智能服务安全基本要求

JR/T 0197—2020 金融数据安全 数据安全分级指南

3 术语和定义

GB/T 45288.1-2025 界定的以及下列术语和定义适用于本文件。

3.1

程序化业务逻辑 Programmatic Business Logic

一种由确定性代码和自然语言指令构成的、可执行、可核验、可扩展的结构化业务流程。

注：一般基于业务流程的SOP转化而成，是该领域最佳实践的知识表达，能够通过精确的程序化逻辑控制结构（如具有确定性的任务分解、步骤、控制流、数据流以及错误处理逻辑等），避免自然语言的歧义与模糊性。

注：可核验是指将复杂的业务流程分解为独立的、可验证的逻辑单元，能够在关键节点对其中间结果的正确性与合规性等进行验证。

注：可扩展是指通过模块化的设计，提供类似编程语言的大规模扩展能力，允许不同的业务逻辑被复用、组合或新增，以支持复杂业务流程的快速构建与迭代。

3.2

场景知识库 Scenario-oriented Knowledge base

与应用场景相关的，经过结构化表示的用于存储领域内的核心概念、实体、属性或相互关系的知识集合。

注：在专业应用领域可以为大模型提供高质量、可验证、可溯源的背景知识，作为事实核查的基准，有效减少模型产生幻觉的风险，并使其推理过程有据可查。

3.3

工具 Tool

由程序化逻辑调用以执行具体操作的一系列经过验证的软件模块集合。

3.4

核验 Verification

在程序执行的关键节点，对操作的合规性、中间结果的合理性、最终输出的准确性以及推理路径的逻辑正确性进行自动化或人工验证的过程。

注：核验是保障大模型应用系统可靠性的核心机制，贯穿于任务执行的全过程。

3.5

核验复杂性塌缩 Verification Complexity Collapse

一个源于计算复杂性理论（特别是NP问题）的核心原理，指对于许多复杂问题，从零开始生成一个正确解的难度（求解复杂性）远高于验证一个给定解是否正确的难度（核验复杂性）。

注：大模型可信应用参考框架利用这一原理，将大模型强大的生成能力与低复杂度的核验能力相结合，可以以较低的成本实现高可靠性。

4 缩略语

下列缩略语适用于本文件。

API：应用程序接口（Application Programming Interface）

NLP：自然语言处理（Nature Language Processing）

SOP：标准作业流程（Standard Operating Procedure）

PBL：程序化业务逻辑（Programmatic Business Logic）

5 概述

本文件中所提出的大模型金融领域可信应用框架，其核心理念是：对于大模型在金融领域所面临的典型风险与挑战，不再仅依赖模型自身的能力提升，而是采用系统工程的思维，将大模型视为一个功能强大但存在一些行为不确定性的核心处理单元，在其周围构建一个确定性的、专业严谨的、可验证的控制与保障体系，从而在系统层面确保最终结果的可信。因此，本文件主要关注大模型的推理应用阶段，并不涉及大模型的开发阶段（如预训练或调优）。

具体而言，该控制与保障体系的核心理念主要包括如下关键环节：

— 业务流程的形式化与结构化。此阶段将金融领域专家的隐性知识和非结构化的业务流程，通过人机协同的方式，转化为一种逻辑清晰、机器可读且可审计的程序化业务逻辑。这一转化过程是对业务流程的精确建模，旨在消除自然语言的模糊性，将复杂的业务任务分解为一系列可独立执行与核验的步骤，为后续的自动化、确定性控制奠定基础。

— PBL的受控执行与过程核验。在业务流程被形式化定义为PBL之后，它将在一个确定性的框架内被调度执行。对于具有较高复杂度的金融业务，在通过意图识别匹配到对应PBL后，通过执行引擎对PBL进行解析并将任务进行合理拆解。对于拆解后的每个子任务，执行引擎精确地编排对上下文数据、工具及模型的调用，并在关键节点对子任务执行结果进行核验，从而确保整个业务流程在系统层面的行为是可预测、可追溯且稳定可靠的。

— 真实业务场景的有效性验证。一个在技术上执行通过的PBL，其输出结果仍需验证是否满足真实金融场景的业务目标。因此，在PBL被批准用于规模化生产部署之前，其最终输出需通过一套面向真实应用场景的业务基准测试进行系统性评估。只有当PBL的执行结果在准确性、合规性、专业性等方面均满足预先定义的业务验收标准，该PBL才被视为验证通过。

— PBL的自动化迭代与演进。为适应金融业务的持续动态变化，可基于已经过业务场景验证的PBL资产，通过派生机制生成、适配或组合出新的PBL，来适用于更新后的生产任务或新的业务场景。这一自动化迭代机制，可以将依赖人类金融专家进行流程重塑和代码开发的工作，转变为自动化的、高效的、低成本的演进过程，从而实现一个能够自我迭代和大规模场景适配的智能应用体系，为大模型在更多、更复杂的金融业务场景中进行可靠、敏捷的规模化部署提供可能。

本文件将主要以PBL的受控执行与过程核验为重点，在第6章给出了应用框架构成，包括主要功能模块和关键流程，并在第7章中分别描述主要功能模块和关键流程的技术实现指南，用以帮助金融机构在大模型应用落地过程参考实现。

6 应用框架

以PBL的受控执行与过程核验为重点，本文件所描述的大模型金融领域可信应用参考框架如图1所示。

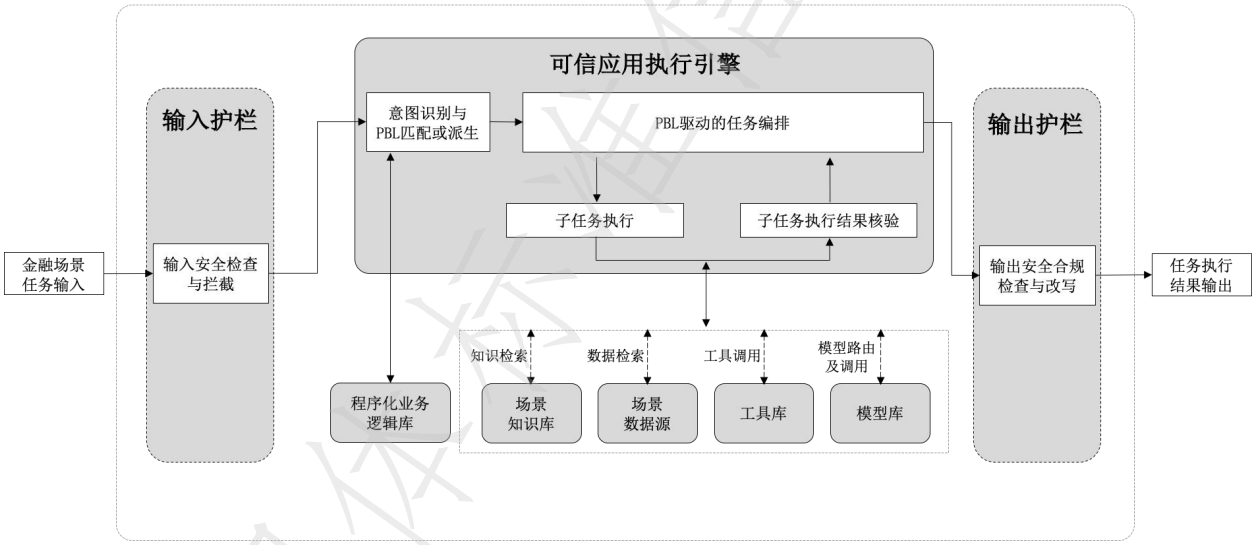

图 1 大模型金融领域可信应用参考框架

如图所示，PBL的受控执行与过程核验流程如下：

— 输入安全检查与拦截：输入金融场景任务处理请求，可选经过输入护栏模块的处理。该模块负责对输入进行安全检查，如识别并拦截恶意攻击、过滤不当内容，并对输入进行标准化处理，保障进入执行引擎的是安全、合规的请求。

— 意图识别与PBL匹配或派生：请求经过安全检查后，会送至可信应用执行引擎进行处理。该引擎是整个框架的核心。首先执行引擎会对任务请求进行意图识别，并从程序化业务逻辑库中检索或生成与意图相匹配的PBL。PBL定义了完成整个任务所需的步骤、逻辑、条件和验证点。

— PBL驱动的任务编排：执行引擎会加载本次场景任务处理请求相对应的PBL，并按照PBL中的逻辑编排完成整个任务所需要的一系列操作，这些操作可能会按照顺序、并行或条件分支的方式进行编排，包括：

- 知识检索：向场景知识库发送结构化查询，以获取精确、可验证的事实依据；
- 数据检索：从场景数据源中检索相关的非结构化或半结构化文档如金融市场实时数据、研报、公告等，作为上下文补充；

- 工具调用：调用工具库中预定义的、确定性的外部工具，执行如数值计算、数据API查询、风险模型调用等专业任务；
- 模型路由及调用：在PBL流程中，在遇到适合大模型处理的任务时如非确定性语义理解或内容生成等，可以从模型库中选择合适的模型并进行调用，由大模型生成相应的任务结果。该步骤可与前面操作相结合，如整合从知识库、数据源和工具执行获取的所有上下文信息，并构建出一个信息丰富、指令明确的提示，输入给大模型进行处理。

— 执行结果核验：大模型的输出结果会返回给执行引擎，并执行相应的核验逻辑。核验过程会利用相应的核验策略来评估大模型输出的准确性、合规性和合理性等。
  - 如果核验通过，执行引擎将继续执行后续步骤；
  - 如果核验失败，执行引擎将根据预设的错误处理逻辑采取行动，例如：重新构建提示并再次调用大模型、调用一个备用模型或者在多次失败后将任务标记为异常并上报人类专家介入等。

— 输出安全合规检查与改写：当成功执行完所有步骤后，执行引擎会负责整合所有中间结果，形成最终的响应。该响应在交付前，可选通过输出护栏模块进行处理，并进行安全合规检查与改写，例如，防止敏感数据泄露、确保内容符合监管要求、添加必要的免责声明等。

7 技术实现指南

7.1 主要功能模块

7.1.1 可信应用执行引擎

7.1.1.1 通用要求

可信应用执行引擎是整个可信应用框架的执行控制中心。其核心职责是解析并执行程序化业务逻辑，对任务的全生命周期进行编排、状态管理和执行结果核验。执行引擎需确保确定性的业务控制流与大模型等非确定性处理单元的分离，将大模型作为受控的处理单元进行调用。

7.1.1.2 意图识别与 PBL 匹配或派生

执行引擎需具备或集成任务意图的准确识别能力，包括：

— 引擎宜能对来自输入护栏的、经过安全检查的标准化请求进行深层语义分析，以理解任务的核心需求、约束条件和最终目标。

— PBL匹配或派生：
  - 对于已经过实际业务场景验证后的PBL对应任务，引擎宜能从程序化业务逻辑库中匹配并加载预置的PBL；
  - 对于新场景任务或者有更新的场景任务，引擎宜能从程序化业务逻辑库中匹配并加载相关联的PBL，并在此基础上基于任务意图动态派生出新的PBL。派生出的PBL宜进行自动化验证或引入人类专家验证，以确保派生PBL在逻辑上是完备和合规的；
  - 当任务意图模糊或存在歧义时，引擎宜具备启动机制的能力，通过与任务请求方的主动交互来消除不确定性，确保后续执行的准确性。

— 宜建立PBL的可信校验机制。在PBL被加载至执行引擎前，宜对其来源的真实性、内容的完整性及代码的安全性（如静态扫描、依赖项检查）进行验证，以防范恶意或被篡改的PBL被执行。

7.1.1.3 状态管理与上下文维持

为支持金融领域中的多步骤、长周期复杂任务，执行引擎需提供一个健壮的状态管理机制，包括：

— 宜为每一个独立的任务请求或业务流程创建一个隔离的执行会话。所有与该会话相关的状态信息，包括变量、中间数据、外部调用结果等，都在此隔离环境中进行管理，避免会话间的数据污染或信息泄露。
— 对于可能中断或需要长时间运行的任务，例如需要人工审批的流程，引擎宜支持执行状态的持久化存储。这确保了任务可以在中断后从断点处精确恢复，而无需从头开始。
— 引擎宜能够在PBL执行的各个步骤之间准确地传递上下文信息，这包括将知识检索、数据检索和工具调用的结果，整合成结构化的上下文，供后续步骤尤其是大模型调用。

7.1.1.4 PBL 驱动的任务调度与编排

执行引擎的核心功能是作为PBL的解释器和调度器，将PBL中定义的逻辑精确地转化为一系列可执行的操作，包括：

— 引擎宜能够解析并执行PBL中定义的各种控制流结构，包括顺序执行、并行处理、条件分支和循环等。
— 为提升处理效率，引擎宜支持对无依赖关系的PBL步骤进行异步和并发调度。引擎宜具备管理这些并发任务执行状态、处理其返回结果并确保最终结果同步的能力。

7.1.1.5 错误处理与容错机制

金融领域的低容错率要求执行引擎需具备错误处理和容错能力，包括：

— 引擎宜能感知来自各功能模块的执行失败信号，并对其进行分类如瞬时性故障、逻辑错误、数据异常、核验失败等。
— 宜支持可配置的超时机制。对于PBL中调用的外部组件（如工具、模型、知识库或数据源），执行引擎宜能为其设置合理的执行超时阈值。当调用超过预设时长未返回结果时，引擎需能主动中断该操作，并将其视为一种执行失败信号。
— 引擎宜支持可配置的重试机制，可允许针对不同类型的错误和任务步骤，定义不同的重试策略，如最大重试次数、重试间隔等。
— 对于关键任务步骤，PBL中宜定义明确的回退（Fallback）逻辑。当主要方法如大模型调用在多次重试后仍核验失败时，执行引擎宜能自动切换到预定义的备用方案或服务降级策略，如调用一个更简单但更稳定的规则模型等。
— 当所有自动化恢复手段均告失败，或任务本身风险等级较高时，执行引擎宜能够暂停工作流，并触发人工介入流程。
— 宜支持配置化的人工介入机制。机构可在PBL生成阶段主动定义人工审批、复核或确认节点，用于在关键业务环节实施人工干预。该节点为可选配置项，支持权限控制、审批时限与决策记录。
— 在人工介入过程中，引擎宜打包完整的上下文信息，包括输入数据、执行轨迹、相关中间结果和错误日志等，并将其推送至人类专家工作台进行处理。所有人工操作与决策结果应记录在可防篡改的审计日志中，包括操作人、时间、审批结论、影响范围等。

7.1.1.6 日志与可追溯性

为符合金融行业的监管和审计要求，执行引擎需生成全面、详细且不可篡改的执行日志，包括：

— 日志宜完整记录从接收请求到输出最终结果的全过程，内容可包括：唯一的交易/请求ID；加载的PBL及其版本；每个步骤的开始与结束时间戳；对外部组件（知识库、数据源、工具、模型）的每一次调用的完整请求与响应数据；模型推理过程中产生的中间推理步骤描述；每一次执行结果核验的输入、所用策略、核验结果及详细理由，以及所感知的错误和采取的恢复措施等。

— 日志宜采用结构化格式如JSON，并存储在可供快速查询和分析的系统中，以支持事后审计与故障排查，并为后续的持续反馈与迭代提供了数据输入。

7.1.1.7 性能与可用性监控

为确保执行引擎作为核心调度服务的稳定性、高效性和连续性，需建立相应的技术运行监控机制，包括：

— 宜对引擎的关键性能指标进行实时度量和监控，指标可包括任务请求吞吐量、并发处理能力、PBL平均执行时长、各关键步骤（如工具调用、模型路由、知识检索）的平均延迟等。
— 宜对引擎服务的可用性进行监控，可包括服务健康状态、心跳检测、以及所依赖的计算和存储资源（如CPU、内存）的利用率和饱和度等。
— 宜建立自动化的异常检测与告警机制。当关键性能指标偏离预设的正常基线，或可用性指标出现异常时，宜能及时触发告警，并通知技术运维团队介入排查。

7.1.2 程序化业务逻辑库

7.1.2.1 通用要求

程序化业务逻辑库是金融机构内的程序化业务逻辑的集中存储、管理和治理中心，其中存放的每一个PBL都代表了一项经过开发和实际业务场景已验证的、可执行的金融业务标准作业流程。逻辑库的设计和管理需确保PBL的完整性、一致性、安全性和可追溯性，为执行引擎提供稳定、可靠的业务流程指令来源。

7.1.2.2 任务拆分与逻辑表达

PBL的核心在于对任务的合理拆分与精确表达，一个复杂的业务任务可被分解为确定性部分和不确定性部分：

— 确定性逻辑：任务中具有明确过程、固定逻辑和稳定判断标准的部分，可使用形式化的程序语言（如Python）进行描述。这部分构成了任务执行的骨架，确保了过程的确定性、可重复性等。
— 不确定性逻辑：任务中涉及模糊概念理解、语义匹配和开放式推理的部分，是传统程序语言难以处理而大模型擅长的领域，这部分可通过自然语言指令来描述。
— 颗粒度控制：对于不确定性部分的任务描述，可进一步分拆并达到大模型能力范围内能够准确执行的颗粒度，以避免因任务规模和复杂度超出模型极限而导致的能力崩溃。

7.1.2.3 资产化管理与生命周期

逻辑库中的每一个PBL需被视为一项关键的软件资产，并实施全生命周期管理，包括：

— PBL宜以结构化的方式存储，清晰分离其核心元素，包括确定性逻辑代码、面向大模型的自然语言指令、核验节点定义、控制流、数据流以及元数据等。
— 每个PBL宜具有明确的生命周期状态，如开发中、审核中、已批准、已验证、和已废弃等。状态的转换宜遵循预定义的治理流程。
— 宜建立PBL的持续评估与迭代机制。宜定期结合生产环境的抽查样本与真实用户反馈，对PBL的执行效果进行度量。基于评估结果，可通过优化PBL中的自然语言指令、调整PBL元数据或调优确定性逻辑等方式，对PBL资产进行迭代优化，以确保持续满足业务需求并修正潜在偏差。

7.1.2.4 版本控制与审计追溯

为符合金融领域严格的审计与合规要求，逻辑库需具备版本控制和审计追溯能力，包括：

— 宜使用版本控制系统对所有PBL进行管理，对PBL的任何修改宜提交为新的版本。
— 宜记录每一次操作的详细日志。日志内容宜包含操作人、操作时间、PBL标识符、版本号以及变更摘要，并确保日志不可篡改，以支持完整的事后审计和责任追溯。

7.1.2.5 模块化与可复用性

逻辑库的设计宜支持PBL的模块化和可复用性，以提升开发效率和系统可维护性，包括：
— 宜将复杂的业务流程拆分为独立的、功能内聚的PBL子模块，如一个客户身份验证或反洗钱筛查的子流程可以被封装成一个独立的PBL模块。
— 宜提供清晰的依赖管理机制，允许一个PBL明确声明其对其他PBL模块、工具库或数据源的依赖关系。这有助于确保流程的完整性，并在依赖项更新时进行影响分析。

7.1.2.6 治理与审批流程

PBL的变更和发布宜经过一个多方审查流程，确保其业务逻辑的正确性和合规性，如：
— 由相关业务部门的领域专家进行业务审查，确认PBL的逻辑与实际业务需求和保持一致。
— 由技术部门进行交叉代码审查，确保代码质量、性能和安全性。
— 由合规、风险管理、个人信息保护、消费者权益保护等职能部门进行联合审查，确保流程设计符合内外部法规和风险控制要求等。
— 对于新版本的PBL，需经过实际业务场景的验证后才可被规模化部署和应用。

7.1.2.7 元数据与可发现性

为了使执行引擎能够高效、准确地根据任务意图匹配到合适的PBL，逻辑库宜提供一个基于元数据的查询接口，允许执行引擎根据意图、场景或其他属性快速检索和定位到相应的PBL。同时，逻辑库中的每个PBL宜附带标准化的元数据以方便查询，包括：
— 唯一标识符（ID）与版本号，用于精确引用。
— 业务名称与描述，用业务语言清晰描述其功能和目的。
— 触发意图/场景，描述该PBL适用的任务意图或业务场景，供意图识别模块匹配。
— 输入/输出定义，明确定义PBL执行所需的输入参数和预期的输出结构。
— 所有者与维护者，明确负责该PBL的业务部门责任人和技术维护方。
— 风险等级，根据PBL所涉业务的影响程度、数据敏感性或监管要求等因素，明确其风险级别。

7.1.3 场景知识库

7.1.3.1 概述

场景知识库是面向应用场景专门构建的结构化知识集合，用于存储领域内的核心概念、实体、属性和其相互关系。它可采用多种技术实现（如知识图谱、关系型数据库、键值存储等），可为大模型提供高质量、可验证、可溯源的背景知识，作为事实核查的基准，可以有效减少大模型产生幻觉的风险，并使其推理过程有据可查。

7.1.3.2 质量评估指标

可建立场景知识库数据质量评估机制，关键指标可包括准确性、完整性、时效性、一致性和可追溯性等。
— 准确性：知识库中信息正确反映客观世界的程度。
— 完整性：知识库中包含目标领域所有相关实体和关系的程度。
— 时效性：知识库中信息保持最新状态的程度。

— 一致性：知识库内部不存在逻辑矛盾，如一个人不能有两个出生日期。
— 可追溯性：知识库中的每个事实都能追溯到其原始来源。

7.1.3.3 查询接口

可通过标准化的接口提供场景知识库的对外查询接口，以支持不同的查询需求：
— 可提供支持结构化查询语言（如SQL、键值查询，或SPARQL、Cypher等图查询语言）的端点，用于精确、复杂的查询。
— 可提供一个面向大模型的查询前端，该前端能将自然语言问题转换为形式化的查询语句（如SQL或图查询），降低大模型直接生成复杂查询的难度。

7.1.3.4 知识库管理

可建立场景知识库的数据管理机制，包括：
— 构建:
  - 从结构化和非结构化数据源构建知识库的过程宜明确且可审计的；
  - 可定义和维护能够准确反映领域复杂性的数据模式或本体；
  - 可从异构数据源（数据库、文档、API）中抽取、转换和融合数据，并解决实体歧义；
  - 可使用大模型辅助进行信息抽取，抽取流程可包含验证和修正环节，以确保最终入库数据的准确性。
— 维护:
  - 可建立清晰的流程来更新、修正和扩展知识库，包括处理知识演变和数据模式变更的机制；
  - 可建立支持领域专家深度参与和持续投入的机制，以应对领域知识的演变。
— 下线：
  - 宜建立知识库数据的下线流程。当数据所支持的业务场景终止、数据已过时或因合规性要求不再适用时，宜有清晰的审批流程启动下线；
  - 下线前宜进行影响分析，确保所有依赖该知识库数据的PBL或服务已解耦或更新；
  - 对于已下线的知识库数据，宜根据数据安全与档案管理规定执行安全删除或归档。

7.1.3.5 安全与访问控制

— 宜实施严格的访问控制策略，访问请求中需携带有效凭证（如工单、权限令牌等），确保操作者身份可追溯，并宜结合业务逻辑与访问数据，动态判断操作是否符合安全策略。
— 宜根据调用方的身份和权限，在知识库的数据单元（如表、条目、字段、子图或节点）层级上进行细粒度的访问授权，确保敏感知识如未公开的投研策略、客户关联网络等，不会被非授权访问。
— 宜记录所有的数据访问请求和查询操作，并确保日志不可篡改，以支持完整的事后审计和责任追溯；
— 知识库的存储、传输和处理宜符合金融行业的数据安全要求，包括但不限于《中国人民银行业务领域数据安全管理办法》等。

7.1.4 场景数据源

7.1.4.1 概述

场景数据源主要存储和管理与应用相关的金融领域非结构化与半结构化数据，如研究报告、公司公告、新闻资讯、法律法规、内部文档等，可以为执行引擎提供丰富、及时、准确的上下文输入数据，作为大模型生成内容的依据，从而提升生成内容的真实性、时效性和深度。

场景数据源与场景知识库的区分在于，知识库提供的是经过提炼的、结构化的事实断言，而场景数据源提供的是蕴含这些事实的原始文本或半结构化文档。

7.1.4.2 数据引入

- 宜建立标准化的数据源引入流程，对内外部数据源（如监管机构网站、数据供应商、新闻通讯社、内部研究平台）进行评估和认证，确保其权威性、可靠性和合法性。
- 宜建立自动化的数据获取与处理流水线，对入库数据进行系统化的质量提升操作，具体包括但不限于数据清洗、数据去重、数据去毒与合规过滤等。
- 宜建立面向高时效性数据（如市场新闻、价格数据、政策公告）的近实时更新机制，确保数据源内容能够及时反映市场动态，满足金融场景对信息新鲜度的要求。

7.1.4.3 内容组织与索引

为实现高效、精准的检索，场景数据源中的内容宜建立精细化的组织和多维度的索引，包括：

- 宜采用智能化的文档分块策略，将长文档切分为语义完整、大小适中的知识块。宜根据文档类型评估并采用不同策略，如基于固定长度的滑动窗口、基于文档结构（如章节、段落）的语义分块，或结合两者优势的混合策略。
- 宜对每个文档及其派生出的知识块附带一组标准化的、结构化的元数据。元数据宜包括：数据来源、发布日期、文档类型、涉及的实体（公司、人物、产品）、核心主题、摘要信息等。
- 宜为知识块构建混合索引，以支持多样化的检索需求如向量索引、全文索引等。

7.1.4.4 数据安全与合规

- 宜遵循金融行业数据安全规范 JR/T 0197—2020，对数据源中的数据进行安全分级，并根据数据级别实施不同的存储加密策略和访问控制权限，确保高安全等级的数据得到相应保护。
- 对数据源的访问请求宜经过严格的身份认证和权限校验，访问请求中需携带有效凭证（如工单、权限令牌等），确保操作者身份可追溯，并宜结合业务逻辑与访问数据，动态判断操作是否符合安全策略。
- 宜记录所有对数据源的访问和数据检索操作，形成详细且不可篡改的审计日志，以满足合规审查和安全事件追溯的要求。

7.1.5 工具库

7.1.5.1 通用要求

工具库是一系列经过验证和管理的外部软件模块、应用程序接口（API）或函数的集合。大模型可以通过调用这些工具来执行具体、确定性的操作，如数据查询、数值计算、文件操作或与第三方系统交互。每个工具都通过标准化的、机器可读的描述文件进行定义，以便大模型能够理解其功能和使用方法。其核心作用是将金融领域中需要精确计算、确定性执行或与外部系统交互的任务，从大模型非确定性的推理中剥离出来，交由经过充分验证的专业工具执行。

大模型对于工具库的调用设计与实施需遵循“白名单”安全原则，即只授权大模型调用白名单列表中的已验证工具，为整个应用系统的稳定性和安全性提供基础保障。

7.1.5.2 工具描述文件

- 宜定义标准化的工具描述文件，描述文件中宜包含工具名称、工具标识、功能描述、版本、参数模式、返回结构、安全上下文等元素。

— 工具库宜提供一个注册中心，存储所有工具的描述文件，并提供基于功能、名称、版本等元数据的查询接口，以便执行引擎或开发人员能够快速发现和集成所需工具。

7.1.5.3 工具的治理与生命周期管理

工具库中的每一个工具需被视为一项受控的资产，实施相应的全生命周期治理，包括：

— 引入与审查流程：新工具的引入宜经过一个多方审查流程，包括技术验证、安全评估和合规审查等。
— 供应商风险管理：当引入第三方提供的工具或服务时，宜进行相应的供应商风险评估。评估应重点关注其数据安全实践、模型透明度、服务等级协议及合规认证情况等。
— 版本控制：宜对所有工具实施版本控制。PBL在调用工具时宜明确指定版本号，以避免因工具升级导致非预期的行为变更。
— 下线管理：宜建立清晰的工具下线流程，包括评估下线影响、通知所有依赖方以及进行安全下线处理等。

7.1.5.4 工具调用方式

可支持多种调用方式，如同步模式、异步模式等。

7.1.5.5 工具调用安全机制

宜定义工具调用和执行所需的安全权限、认证方式和数据处理策略等，以确保工具调用过程的安全，包括：

— 建立相应的工具调用身份认证和访问控制机制，每一次来自执行引擎的调用宜携带可验证的凭证（如工单、权限令牌等），确保操作者身份可追溯，并宜结合业务逻辑，动态判断操作是否符合安全策略。
— 工具的调用接口宜对所有输入参数进行格式和内容验证，以防范注入攻击等安全威胁。
— 工具的输出在返回给执行引擎前，宜经过过滤和审查，确保不会泄露非授权的敏感数据。

7.1.5.6 工具监控与审计

为符合金融行业的合规与可追溯性要求，对工具库的所有操作和调用宜进行全面监控和记录，包括：

— 宜建立实时的监控机制，追踪每个工具的性能指标，如调用频率、平均延迟、错误率和系统可用性。当指标偏离正常阈值时，应能触发告警。
— 宜为每一次工具调用生成详细且防篡改的审计日志。日志内容宜包括：调用时间、调用方身份、访问凭证、被调用工具及版本、完整的输入参数和返回结果。

7.1.6 模型库

7.1.6.1 概述

模型库是可信应用参考框架中相关人工智能模型（包括通用大模型、金融领域微调模型、以及其他专有模型）的集中化、版本化、资产化管理中心，可以为执行引擎实现模型路由、选择和交叉核验策略提供支持。模型库的建设与运营需确保其中每一个模型资产的来源可追溯、性能可量化、风险可管理、使用可审计。

7.1.6.2 模型准入

任何模型在被纳入模型库并提供服务之前，宜通过相应的准入流程，包括：

— 入库前宜经过来源验证、安全扫描和审批流程，以防范软件供应链安全风险。

— 宜对模型能力进行较为充分全面的评测验证，以量化其在准确性、幻觉率、偏见性等关键指标上的表现。
— 每个入库的模型版本宜配备相应的模型卡，该文档是模型透明度的核心载体，宜包含以下信息：
  - 模型的架构、参数规模和训练方法；
  - 训练数据集的来源构成、时间范围；
  - 在关键评测基准上的性能指标；
  - 已知的局限性、潜在的偏见风险和可能产生幻觉的场景；
  - 明确的、经过批准的预期应用场景和禁止使用的场景等。
— 宜对模型进行系统的安全风险评估，识别其在提示注入、模型投毒、敏感信息推断、生成内容安全合规性等方面的脆弱性，并制定相应的缓解措施。

7.1.6.3 模型版本与生命周期管理

宜对所有模型资产实施全生命周期的版本化管理，以应对模型的快速迭代和金融业务的演进需求，包括：

— 宜对每个模型版本都分配唯一的标识符。PBL在请求模型服务时，宜能指定具体的模型及其版本。
— 每个模型版本宜具有明确的生命周期状态，如评估中、使用中、已弃用等。状态的变更宜建立管理流程，并对变更理由进行记录。
— 宜维护一个清晰的依赖关系图，记录哪个版本的PBL或应用正在使用哪个版本的模型，这对于模型版本更新或弃用时的影响分析较为重要。
— 宜支持快速、可靠的模型版本回滚机制。当新上线的模型版本在生产环境中表现出非预期的行为或性能下降时，执行引擎宜能被配置为迅速切换回上一个稳定的版本。

7.1.6.4 模型路由与选择

模型库宜提供支持动态路由与选择的能力，包括：

— 模型库及其接口的设计宜能支持执行引擎实现灵活的路由策略，如执行引擎可根据任务的优先级、成本预算或对延迟的敏感度，从模型库中动态选择最合适的模型。
— 模型库的架构宜能够支撑“正向交叉核验”等高级核验策略，即它宜能同时向执行引擎暴露多个功能相似但实现不同的模型服务，以便执行引擎可以并行调用并比对它们的输出结果。

7.1.6.5 安全与合规

— 模型的权重文件、配置文件和训练代码等核心知识产权和高价值资产，宜采取严格的安全措施进行保护，包括静态加密存储、严格的访问控制和防泄露机制等。
— 对外提供模型服务的推理端点宜得到充分的安全加固，包括实施身份认证和授权、API调用频率限制、输入内容过滤等措施，防止模型被滥用或遭受拒绝服务攻击。

7.1.6.6 监控与性能评估

— 宜对模型库中所有使用中的模型推理服务进行7x24小时的运行状态监控，关键指标包括：推理延迟、吞吐量、错误率和资源利用率等。
— 宜建立自动化的机制，定期或在检测到数据分布变化时在标准评测集上重新评估线上模型的性能，以检测可能出现的性能漂移。性能的显著下降宜触发告警和重新训练/微调流程。
— 模型的性能监控宜与业务指标和核验度量指标等相关联，如持续追踪特定模型在特定PBL任务中的核验通过率等。

— 当模型指标出现异常的情况超出预设阈值后，宜自动熔断该模型并切换至备用模型或者触发人工介入流程，并将熔断事件写入日志。

7.1.7 输入护栏

7.1.7.1 概述

输入护栏是可信应用参考框架的安全防范入口，外部输入（包括任务请求和系统API调用）在进入“可信应用执行引擎”之前，可经过此模块的处理。其核心职责是识别、拦截并净化潜在的恶意、有害或不合规输入，同时对合法输入进行标准化处理，保障进入核心系统的请求是安全、规范且可预测的。

7.1.7.2 安全威胁防御

输入护栏宜具备识别和抵御针对大模型应用的常见安全威胁的能力，包括：

— 宜实施有效的机制来检测和防范提示注入攻击，具体如：
- 指令与数据分离：采用技术手段（如分隔符、输入重构）将输入的数据与系统内部的指令模板分离，降低输入被解释为恶意指令的风险；
- 攻击模式识别：宜综合利用复杂的规则集（用于识别已知的注入攻击模式）与具备语义分析能力的安全模型（用于识别新型、对抗性或可能的注入模式），对输入进行分层检测，以识别如指令覆盖、角色扮演诱导等攻击；
- 输入净化：对检测到的可疑输入进行无害化处理，例如移除或转义潜在的指令性关键词，然后再传递给下游模块。

— 宜集成内容安全检测能力，对输入进行扫描，并自动过滤或拒绝不合规的内容（如涉黄、涉暴、涉政等），以及拒绝与金融业务场景完全无关的、可能导致模型输出不当内容的请求。

— 宜具备针对非结构化文件（如PDF、Office文档、图像、压缩包等）的安全扫描能力。宜能进行恶意软件检测、病毒查杀、宏代码分析与文件完整性校验，以防范嵌入式恶意代码执行、或文件解析漏洞利用等攻击。

— 宜实施基本的资源滥用防范措施，以保护系统免受拒绝服务（DoS）类攻击，包括：
- 请求速率限制：对来自同一来源（如IP地址或用户ID）的请求频率进行限制；
- 输入长度限制:对输入请求的长度设置合理的上限,防止因处理超长输入而耗尽系统资源；
- 指令内容安全分析：宜建立机制以分析指令内容本身可能引发的计算资源滥用。例如，识别并限制可能导致模型进行过度递归、深度迭代或不合理长时间推理的特定指令模式，以防范针对模型推理引擎的拒绝服务（Model DoS）攻击。

7.1.7.3 任务合理性检查

输入护栏宜执行任务合理性检查。宜能够根据预设规则，判断任务请求是否属于系统定义的服务范畴。对于明显超出业务范围的请求，应予以拒绝。

7.1.7.4 输入标准化与预处理

输入护栏可执行标准化与预处理操作，为可信应用执行引擎提供格式一致、易于处理的输入，包括：

— 将不同来源、不同格式的原始输入统一转换为系统内部标准的结构化格式如JSON，包括去除不必要的字符、统一编码等；

— 在多轮对话场景中，输入护栏可负责安全地加载和管理历史对话。在将历史对话与当前输入合并时，宜对历史内容同样应用上述的安全与合规检查，防止历史对话中包含的恶意内容被重新激活。

7.1.7.5 日志与监控

— 宜以结构化、防篡改的方式记录通过输入护栏的请求及其处理结果。对于被拦截或修改的请求，日志中宜详细记录原始输入或其哈希值、触发的规则、拦截原因和时间戳，以备安全审计和事件溯源。
— 宜持续监控输入护栏的运行状态和性能指标，包括请求处理的吞吐量、延迟、各类威胁的拦截率等，并建立告警机制。

7.1.8 输出护栏

7.1.8.1 概述

输出护栏是可信应用参考框架的质量与安全控制出口，预备交付给用户或下游系统的结果需经过此模块的安全处理。其核心职责是在输出前进行一道安全、合规与适当性审查，确保交付的内容是安全的、负责任的、符合监管要求的，并对潜在问题进行拦截或安全改写。

7.1.8.2 安全与合规审查

输出护栏宜执行相关审查以识别并处理不符合安全与合规要求的输出内容，包括：
— 宜实施敏感数据扫描机制，以检测并阻止任何形式的数据泄露，包括：
  - 能识别各类金融敏感信息，包括但不限于客户非公开信息（NPI）、个人身份信息（PII）、账户详情、交易记录、以及机构内部的自营交易策略、模型参数或PBL代码片段；
  - 根据预设的数据安全策略，对检测到的敏感信息执行相应的处理动作，如完全拦截、部分或全部脱敏（遮蔽），或替换为通用占位符等。
— 宜内置一个可配置的、动态更新的金融监管规则引擎。对于客户沟通内容（如营销材料、投资建议等），宜检查其是否遵循相关法规，确保内容公平准确，无误导性或夸大性陈述等。
— 宜自动检查并添加法规要求的强制性披露信息，如免责声明、风险提示、投资顾问资质信息等。
— 宜对大模型生成的内容进行有害信息扫描，过滤和拦截任何不当输出，包括仇恨言论、歧视性语言以及其他任何可能损害用户或机构利益的内容等。

7.1.8.3 输出改写与格式化

当检测到轻微的合规或安全问题时，护栏可尝试进行改写，而非直接拒绝，如：
— 自动为缺乏风险提示的投资观点补充标准的风险警示语。
— 将过于绝对的表述修改为更合规的、中性的措辞。
— 输出护栏可负责对最终输出进行格式化处理，确保其风格、语调和品牌呈现的一致性，如统一日期格式、货币符号，或确保所有输出采用机构官方的问候语和结束语等。
— 可在最终输出中附加必要的溯源与透明度信息，增强用户的信任感，如标注关键信息或数据的来源等。
— 在人工智能生成合成内容标识上，应符合GB/T 45438-2025的要求。

7.1.8.4 日志与监控

— 宜以防篡改的方式，记录进入输出护栏的原始输出和经过处理后的最终输出。对于所有被拦截或修改的响应，日志中宜详细记录触发的规则、执行的动作（拦截/改写）以及时间戳等。
— 宜持续监控输出护栏的各项关键指标，如总请求量、拦截率、各类规则的触发频率、改写率等。这些指标的异常波动宜能触发实时告警，提示相关团队可能存在上游模块如模型或PBL的变更风险或性能衰退问题。

7.2 关键流程

7.2.1 输入安全检查与拦截

此流程的目标是在进行业务逻辑处理之前，对传入的原始请求进行系统性的、多层次的审查和净化，以识别并阻断潜在的安全威胁、合规风险和不当内容，确保进入后续处理环节的请求是经过验证的、安全的、标准化的。

对于直接面向外部用户、合作伙伴或部署在非可信网络环境的应用场景，宜支持本流程。对于部署在金融机构内部可信域、且输入源自其他已认证内部系统的应用，可根据机构的整体安全架构和风险评估结果，自行决定是否支持本流程。

流程一般包括：

— 步骤一：请求接收与标准化
  - 接收来自不同渠道的原始请求，并转换为系统内部统一的编码和数据格式，消除因编码或格式不一致可能引入的解析漏洞；
  - 清除输入中不可见的控制字符、多余的空白符或与内容无关的标记语言，形成一个干净、规范的文本流，为后续的语义分析做准备。
— 步骤二：安全威胁分析
  - 静态模式匹配，如恶意特征库扫描、违禁内容筛查等；
  - 语义与行为分析，如提示注入意图识别、对抗性攻击模式检测等；
  - 上下文关联分析，如会话历史一致性检查等。
— 步骤三：任务合理性检查
  - 基于预定义的业务规则，判断任务请求是否属于应用提供的服务范畴，如一个用于市场分析的投研助理应能拒绝处理个人银行业务查询。
— 步骤四：处置与拦截决策
  - 根据前述步骤的分析结果，流程将对请求做出最终处置决策，如放行、净化或者拦截等；
  - 对于被拦截的请求，宜向请求来源返回一个标准的、不包含任何内部细节的拒绝服务响应，同时生成安全告警，并在安全日志中记录完整的事件快照，包括请求的哈希值、来源IP、用户标识、触发的规则详情以及处置决策等。

7.2.2 意图识别与 PBL 匹配或派生

此流程的目标是将经过安全净化和标准化的任务请求，转化为一个结构清晰、逻辑完备、参数明确、可供执行引擎直接调度的程序化业务逻辑。它作为连接自然语言意图与系统确定性执行的桥梁，其执行的精确度和可靠性决定了后续所有任务编排的正确性和有效性。

流程一般包括：

— 步骤一：深度意图理解与参数抽取
  - 对请求进行深度的语义分析，以精确识别任务的核心业务意图；
  - 在识别意图的同时，从请求中提取执行任务所需的关键实体和约束参数。
— 步骤二：PBL模板检索与匹配
  - 使用上一步识别出的核心意图和关键实体类型作为查询条件，向程序化业务逻辑库发起检索请求；
  - 在逻辑库中查找与查询条件在元数据层面（如业务名称、触发意图）匹配的、预先定义好的PBL模板。
— 步骤三：PBL派生

  当标准库中不存在现成的PBL模板时，流程宜具备动态派生新PBL的能力，包括：

- 通用SOP适配：即从逻辑库中选择一个与任务意图最相关的通用标准作业流程或基础PBL模板；
- 上下文驱动的定制化：可利用大模型将通用模板与当前请求的具体参数和约束条件相结合，进行定制化生成；
- 生成后验证：所有动态生成的PBL片段或完整PBL，在进入下一步前，宜经过自动化验证或人类专家的验证，确保其语法正确、逻辑闭环且符合机构预设的安全与合规策略。

— 步骤四：PBL实例化与参数化

- 将与请求相关的上下文信息注入到PBL实例的初始状态变量中；
- 完成填充后，形成一个完全实例化的、包含所有必要信息、可被执行引擎直接加载和运行的PBL对象。

— 步骤五：执行前确认与日志记录

- 对于高风险或高价值的业务操作，在将PBL实例提交执行前，系统宜根据PBL的元数据评估其风险等级。对于达到预设风险阈值的操作，宜启动人机交互确认流程，向任务请求方清晰地展示即将执行的操作摘要，并获取其明确授权；
- 宜详细记录本次意图识别与PBL生成流程的完整轨迹，包括原始请求、识别出的意图与参数、匹配或生成的PBL模板ID及版本、最终实例化的PBL内容等，作为后续任务审计、问题排查和系统迭代优化的依据。

7.2.3 PBL 驱动的任务编排

此流程的目标是将静态的、声明式的PBL转化为一个动态的、受控的、可观测的执行过程。在此流程中，执行引擎作为总调度中心，遵循PBL中定义的业务逻辑、控制流和数据流，精确地编排对场景知识库、场景数据源、工具库和模型库等核心能力组件的调用，从而完成复杂的金融任务。

流程一般包括：

— 步骤一：PBL加载与执行上下文初始化

- 加载PBL实例：执行引擎加载在7.2.2流程中最终生成并确认的PBL实例；
- 解析执行计划：引擎对PBL的结构进行深度解析，识别出任务的主干逻辑、所有预定义的执行步骤、控制流（顺序、并行、条件分支）、数据依赖关系以及预设的核验节点；
- 初始化执行上下文：创建一个隔离的、与本次请求唯一对应的执行会话。将PBL实例中定义的初始变量、从任务请求中提取的参数以及必要的会话历史注入到这个上下文中，作为任务执行的初始状态。

— 步骤二：任务调度与资源调用

- 执行引擎根据PBL的指令，按部就班地调度执行一系列原子化或复合化的任务，包括知识检索、数据检索、工具调用、模型路由及调用等；
- 当PBL步骤要求获取确定性的事实依据时，执行引擎根据PBL中定义的查询逻辑，向场景知识库发送结构化的查询请求，并将返回的准确事实更新到执行上下文中；
- 当PBL步骤需要非结构化的背景信息时，执行引擎启动向场景数据源发起检索请求，获取与当前任务最相关的数据块，作为后续模型调用的上下文补充；
- 当PBL步骤要求执行一个确定性的计算或操作时，执行引擎从工具库中调用一个指定版本、功能明确的工具。引擎严格按照工具的API规范封装输入参数，并处理其返回的确定性结果；
- 根据PBL中指定的模型能力要求或路由策略，从模型库中选择合适的模型，并将构建好的提示词发送给该模型进行处理。

— 步骤三：并发与依赖管理

- 执行引擎宜分析PBL的依赖关系图，识别出没有前后依赖关系的独立任务分支，并对其进行并行调度，以最大化执行效率；
- 对于存在依赖关系的任务步骤，执行引擎宜确保前置任务已成功完成且其输出已通过执行结果核验，才能启动后续任务。

— 步骤四：中间结果聚合与状态更新

- 执行引擎宜持续监听所有已调度任务的执行状态和返回结果；
- 一旦任务成功完成并通过核验，其输出结果将被合并到当前会话的执行上下文中，作为后续步骤的输入。这个持续更新的上下文构成了整个任务执行过程的短期记忆。

— 步骤五：执行日志与轨迹追踪

- 在任务编排的每一步，执行引擎宜生成详细的、结构化的、带有时间戳的审计日志。日志宜包括每一次对外部组件的调用、每一次控制流的决策以及每一次执行上下文的状态变更等；
- 这些日志共同构成了一条清晰、完整、不可篡改的执行轨迹，可作为后续执行结果核验和输出安全合规检查进行判断的依据，也是实现系统可解释性、可审计性和可追溯性的保障。

7.2.4 执行结果核验

核验是可信应用参考框架的核心流程，其有效性建立在“核验复杂性塌缩”原理之上，即验证一个解的正确性通常比生成这个解的复杂度低很多。在可信应用参考框架中，核验不是一次性的事后检查，而是一个贯穿于PBL执行全过程的、持续的、多层次的活动，通过复合核验，可以在错误扩散前将其识别并隔离，确保只有经过验证的、可信的中间结果才能被用于后续的任务步骤中，从而保障整个业务流程执行结果的最终可信。

流程一般包括：

— 步骤一：核验触发与上下文准备

当可信应用执行引擎完成一个在PBL中被标记为需要核验的步骤后，本流程被自动触发。执行引擎聚合所有与本次核验相关的上下文信息，形成一个完整的核验任务包，一般包含：

- 待核验对象：刚刚生成的原始输出结果；
- 核验依据：触发该步骤的输入、PBL中为该核验节点定义的具体核验策略及参数；
- 执行轨迹：当前任务的链路执行上下文，包括先前步骤已验证的中间结果等。

— 步骤二：核验执行

根据任务的风险等级和性质，框架可支持并组合使用以下多种核验策略：

- 确定性核验：通过预定义规则或者可直接使用代码进行的核验，例如，“社保领用人员的年龄应该在0-150岁之间，否则为异常”；
- 逆向核验：利用大模型生成的输出结果，反向推导或预测任务的原始输入条件，如果推导出的输入与实际输入一致，则认为输出在逻辑上是自洽的；
- 正向交叉核验：对比多个大模型输出结果的一致性来进行核验的过程；
- NLP要点核验：利用自然语言处理技术，提取输出内容中的关键信息点，并逐项进行比对核验等，例如，比对生成内容与源内容之间的语义一致性或情感倾向等，以确保摘要、翻译或内容改写的忠实度；
- 基于外部工具的核验：调用外部的、专业的验证工具（如代码静态分析工具、数学求解器、合规检查API）对生成内容进行的核验；
- 基于业务自定义规则的核验：根据领域专家结合具体业务场景所自定义的验证规则进行的核验。

— 步骤三：核验结果判定与处置

执行引擎综合所有已执行的核验策略结果，做出最终判定，如通过或者失败。

- 若通过，将已验证的结果正式写入当前任务的执行上下文中，并授权执行引擎继续执行PBL的下一个步骤；
- 若失败，立即中止当前执行路径，阻止不可信的结果污染后续流程，并转入步骤四。

— 步骤四：失败恢复与升级

根据PBL中定义的错误处理逻辑，执行引擎尝试自动化恢复：

- 重试：重新执行失败的步骤，可能会附带修正后的指令，如在提示词中明确指出上一轮输出的错误并要求改正。重试次数和间隔由PBL中的策略决定；
- 回退：若重试达到上限后仍失败，可执行预定义的回退方案，如调用一个更简单但更可靠的工具，或返回一个安全的默认值；
- 人工介入升级：当所有自动化恢复措施均告失败时，系统宜暂停该任务，并将完整的失败上下文（包括执行轨迹、待核验对象、所有核验步骤的失败详情）打包，并推送给相应的人类专家进行最终裁决和处理。

— 步骤五：核验度量与日志记录

无论核验成功与否，核验流程的每一步骤、每一次比对、每一个判定结果都宜被详细、防篡改地记录到审计日志中以确保可追溯性，并宜通过统计看板等展示形式进行后续分析改进。

流程执行后，宜更新与该PBL及所用模型相关的核验度量指标，可使用的指标包括：

- 完成率：大模型能够通过核验输出结果的比率；
- 正确率：在通过核验的情况下，输出结果被判定为正确的比率。

7.2.5 输出安全合规检查与改写

此流程的目标是在PBL全流程执行完毕、响应已生成后，对其输出结果进行一次全面的的审查，以识别并修正任何残余的安全风险、合规瑕疵或不当表述，确保交付的输出是安全、合规且专业的。

对于直接面向外部用户、合作伙伴或部署在非可信网络环境的应用场景，宜支持本流程。对于部署在金融机构内部可信域、且输入源自其他已认证内部系统的应用，可根据机构的整体安全架构和风险评估结果，可选支持本流程。

流程一般包括：

— 步骤一：最终响应接收与解析

- 从可信应用执行引擎接收已完成业务逻辑处理、整合了所有中间结果的最终执行结果；
- 对最终执行结果进行解析，将其分解为独立的、可审查的组成部分，例如主文本内容、数据图表、引用的事实列表、建议的操作等，为后续的分项检查做准备。

— 步骤二：多维度审查

- 敏感数据泄露扫描：对最终执行结果的全部内容进行最后一次扫描，以捕获任何可能在生成过程中被无意引入的敏感数据，如客户非公开信息、个人身份信息、内部信息等；
- 金融监管合规性审查：将最终执行结果内容与一个内置的、可动态更新的金融监管规则库进行匹配检查，检查内容可包括内容公平性与平衡性（确保对产品或服务的描述不存在偏颇、不夸大收益、不淡化风险等）、禁止性承诺检查（筛查是否存在任何形式的投资回报承诺或保证性陈述等）、适当性原则符合性（检查建议是否与目标用户的风险画像存在明显冲突）、完整性检查（验证是否包含了所有法律法规要求强制披露的信息，如完整的免责声明、风险警告、投资顾问的执业编号等）；
- 品牌与声誉风险审查：分析最终执行结果的整体语调、风格和措辞，确保其符合机构的品牌形象和专业标准，不存在可能引发公众误解或损害机构声誉的表述等。

- 内容安全审查：对最终执行结果进行有害信息扫描，识别、过滤或拦截任何不当内容，确保输出内容符合法律法规与社会公序良俗要求。

— 步骤三：处置决策与智能改写

- 根据审查结果，流程做出处置决策，如放行、改写或者拦截等。宜优先采用修正而非直接拦截的策略；
- 当触发了改写决策时，可根据需要进行如下改写操作：追加式改写（对于缺失必要信息的情况，自动在执行结果的适当位置追加内容，如若缺少免责声明，则在末尾附加上完整的、经过法务审核的标准免责声明文本）、替换式改写（对于存在不当表述的情况，进行精确替换）、脱敏式改写（对于检测到的敏感信息泄露执行脱敏处理，如将内部项目代号替换为“某项内部计划”等）；
- 当触发了拦截决策时，宜立即中止结果输出流程，生成一个合规或安全事件告警并通知相关的人工审核团队。同时，在审计日志中详细记录被拦截的响应内容及拦截原因，供事后分析。

— 步骤四：最终格式化与交付

- 对通过审查或经过改写的响应内容进行最后的格式化处理，确保其在字体、排版、日期格式等方面与机构的官方出品保持一致；
- 将最终版本的、安全合规的响应交付给用户或下游系统。同时，将原始响应、所有检查与改写操作、以及最终交付的响应内容完整地记录到不可篡改的审计日志中。