Solution Architecture

For

[IFMS 3.0]

Document Version: [V 1.4]

Date: [25/04/22]

Contents

S. No.	Description	Version	Date	Remarks
1.	Basic Draft Released	V 1.0	26/January/2022	First draft released
2.	Changes Suggested by client and other Stakeholder's feedback addressed.	V 1.1	28/February/2022	Changes suggested by client on physical solution document addressed. Functional missing information updated. Data backup and recovery section added.
3.	Changes suggested by PSF Sir incorporated	V 1.2	25/March/2022	Solution Architecture updated and reviewed by Oracle team
4	Microservice Core Architecture included	V 1.3	01/April/2022	Microservice Core Architecture, API security, Solution context updated
5	Infra Details Updated	V 1.4	25/April/2022	Oracles inputs on monitoring tool, infra details and other open points included

1. Project Background

1.1. Key aspirations for Solution

The following are the aspirations considered when conceptualizing the IFMS 3.0 solution.

- Eliminate/ reduce of human intervention in all processes.
- Standardization including implementation of SSO, e-Vault, Janadhaar
- Enhancement of Facia (UI/ UX)
- Implementation of a common disbursement engine
- Encompass all financial transactions of the state
- Effective integration among various systems

1.2. Areas of Improvement

The IT Modernization has created a technology foundation for the IFMS 2.0 project and has significantly improved the operational efficiency for the department. With the growing needs of IFMS businesses and functions, like any other organization and entity, there is a scope for the second wave of technology transformation, focusing on key improvement areas around functional, technological, and business processes, few of which are provided below.

Functional Improvement Areas

1. There is a need to find out gaps, where process automation is possible and implement that.

2. **Common disbursement engine**: Presently disbursal of payments is happening through disparate/ multiple systems (Pay manager and PRI Pay manager, along with integration with other systems and processes such as RajKosh and Ways and Means). There should be one common disbursement engine for all payments across the state government bodies including parastatals, PSUs, Corporations, etc. for all types of payments like salaries, pensions, vendors, beneficiary payments, etc. with proper built-in validations. Common disbursement engine can also facilitate introduction of payment SLAs and automated prioritization for various expenditure categories and facilitate in determining the cash flow requirements over a short- and medium-term basis across the state, which can support the Finance Department in adequately provisioning for the cash requirements through short term borrowings.

3. **Standardization**: This shall be done for the following:
   • Leverage common application infrastructure created by the state government departments like the DoIT & C/ RISL such as Single Sign-on, e-Vault as a document repository, e-Sign, etc.
   • The IFMS architecture should be state of art and enable seamless integration with other departmental applications, Common IT/ application infrastructure and third-party applications.

   • Ensure adequate security and validations at all levels.

E.g.: State has developed multiple applications for IFMS. These should now work on a single architecture and should all be migrated to IFMS 3.0 with standard microservices based architecture platform.

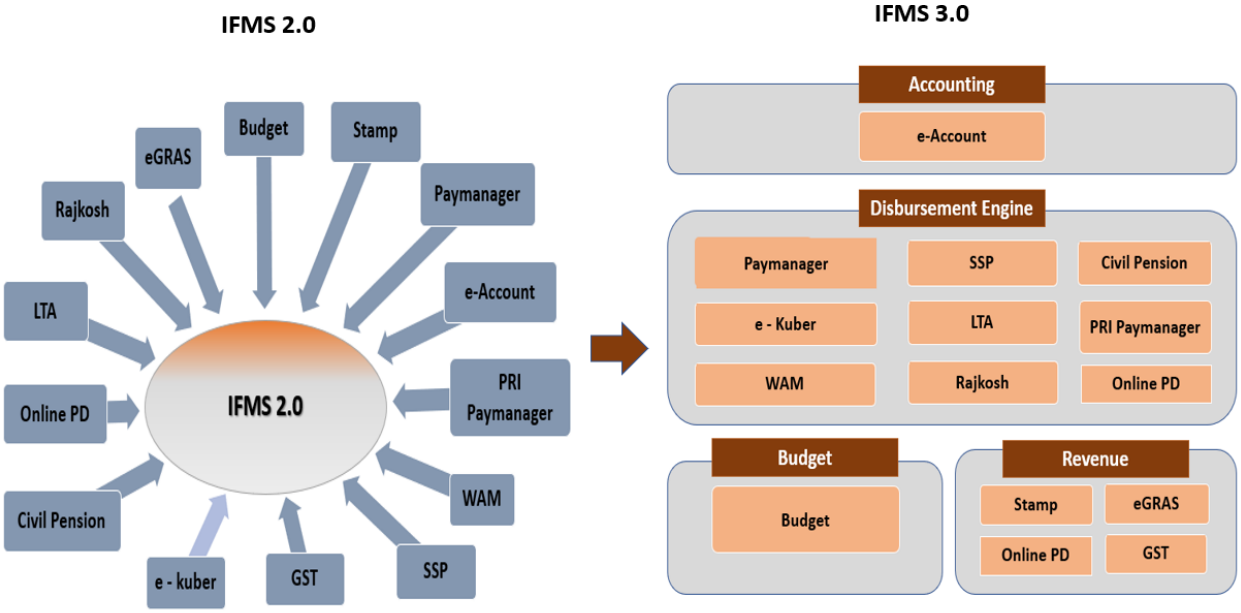
4. Whenever there is a change in requirement or new process, existing IFMS 2.0 has challenges and this needs to be optimized.

5. Role based mapping of all users, IFMS2.0 is scattered across multiple modules and roles, there is a need for centralized login and SSO implementation.

6. Document management System to be optimized and be managed efficiently.

7. IFMS Shall encompass all financials transactions of the state including those of parastatals, boards, corporations, ULB's etc. for all kinds of revenue receipts and payments.

Facia (UI/ UX): UI/ UX of the system must be enhanced to ensure state of art and user-friendly UI on all IFMS interfaces/ channels including portals, mobiles, etc. These should have best of features that are available in the most popular portals globally. Presently there are challenges related to user experience, including multiple logins across various system to execute a single transaction. For example, to make a payment for a single bill and submit the accounts to AG, Treasury Officer is required to login to multiple systems, including digitally signing multiple times - Pay manager, RajKosh and RajORS.



IFMS 2.0 component's mapping with IFMS 3.0 Modules

Technology Improvement Areas

1. Transactional and operational reporting databases may be de-linked, and data archival and retrieval processes may be enabled to reduce load on the transactional database and address performance related issues
2. Implementation of Data Analytics to provide better insight of data.

3. Enabling API and microservices based integration is a key area which needs to be addressed. This is imperative to ensure a seamless integration with third party applications and services, e.g., with banks and other relevant systems.
4. Microservice level API's will be exposed to partners for specific functionalities.
5. Performance of the system to be improved.
6. Systems should be more scalable and configurable.
7. UI/UX Enhancement for more user-friendly experience
8. Implementation of latest technologies and framework to implement more efficient, secure and transparent system.
9. Mobile Application Integration and channel enablement.
10. SNA implementation on IFMS 3.0. (Single Nodal Account)
11. Integration with external stakeholders including banks, RBI, AG, Department, PSU, PFMS, GSTN etc.
12. Provide Access to vendors for bill submission, e-Sanctions etc.

Business Expectations

In addition to the functional and technical improvement areas, there are certain business needs and expectations too which needs to be addressed while designing the overall technology solution for IFMS operations. Few of these expectations are listed below.'

1. Data gathering for proactive planning: Capturing and organizing data for each and every operational task and process as well as for key performance and utilization parameters at each step of the operational chain is imperative for efficient planning, predictive modelling as well as for risk mitigation.
2. Detailed view of transactions with key Partners: The systems should provide a holistic view of funds to authorized person via implementation of funds pool across relevant offices in order to provide better control to the department to plan and manage their activities.
3. There is a need to study current workflows, preparation of current workflows and new workflows (process wise) with specifying new insertions and gaps.
4. Need for amendments in rules for better coordination and working.
5. Close coordination of all technical teams for migration of IFMS 2.0 to IFMS 3.0 in respect of data and core processes.

2. Architecture Principles

The following Architecture principles and design considerations will guide the overall design and implementation of IFMS 3.0:

2.1. Primacy of User Experience

In the multi-stakeholder environment that IFMS operates in, the further success of the platform would be driven by the excellence and experience of the stakeholders/ users on the IFMS technology-enabled service delivery platform. The user experience would be at the heart of the design of the new platform. Design thinking principles would be leveraged to factor the needs and desires of the various stakeholders into developing the IFMS 3.0 platform.

2.2. Omnichannel Workflows

Users of the IFMS 3.0 platform shall be able to leverage multiple access channels, as applicable to the service, to complete the workflow touchpoint interactions. IFMS 3.0 shall enable users to access the platform services through multiple delivery/access channel's role based and the users should be able to conveniently switch across channels to complete a multi-step workflow as per the convenience and choice of the user.

2.3. Mobile First, Self Service & Assisted

In today's time, mobile has become the most-used channel for accessing information and services. Mobile channels are mandatory for the delivery of all services, among all delivery channels in the IFMS 3.0 platform. The design of all services on IFSM 3.0 has to be done on the principles of enabling mobility and then extended to other access/ delivery channels. Enablement of services/ part of workflows in services shall be decided based on business needs as and when required.

2.4. One User One Identity across Enterprise (Multiple Roles)

User Identity and access management definition would be pivotal for the IFMS 3.0 platform to secure access to IFMS data and capabilities and ensure a seamless experience through a single identity per individual. Each user will have a single identity to authenticate himself/ herself. Access to IFMS services and data to any internal/ external user would be extended on top of this identity by assigning them various roles as required, essentially enabling role-based

access control (RBAC). A single user may be granted various roles (as required) under the same identity. However, separation of duties will be enforced so that a single user has clear segregation of activities and permissions even across multiple roles. For example, a user cannot be both maker and reviewer or approver for the same business case or workflow. If additional changes have been officially allocated this may be facilitated through a single administrator user.

While a single identity will be managed for each user, multiple authentication mechanisms shall be enabled to provide multi-factor authentication. Multi-factor authentication will be enabled for users depending on the criticality of the business use case.

2.5. Anytime Anywhere Access

In today's time all organizations have adopted the ability to work from the office and on the move as well, to increase performance and efficiency. Extending on to the principle of Mobile-first, this principle mandates the design of all services and capabilities of IFMS 3.0 to ensure enablement of remote access to all information and services with requisite security controls being in place.

2.6. Maximize Re-Use

Duplicative capability is expensive and proliferates conflicting information. It also makes enhancements to these capabilities difficult to implement. Reusable capabilities enable faster time to market, lower overall costs, and better consistency of capability, processes, and operations. IFMS 3.0 platform would be built using common use applications across the lines of business as shared enterprise standard applications. These applications will serve as the building blocks to enable the core business solutions/ applications across all LoBs against duplicating these in the core business solutions.

2.7. Microservices Based

IFMS 3.0 platform shall embrace a microservice architectural style that sets out to develop applications that are built as a suite of granularly defined services that are independently deployable, each running in its process and communicating through lightweight mechanisms. Each service will encapsulate functionality that is atomic, independent, and reusable, with the ability to use it in diverse contexts.

2.8. Platform/ Stack Approach – Diversity with Interoperability

IFMS 3.0 technology landscape will be built on the concept of an integrated/unified platform of systems (vs. a standardized uniform system). IFMS 3.0 platform with its standards and shared infrastructure will ensure that diversity can co-exist thus enabling various stakeholders to have the flexibility to bring in required solutions/ customizations while creating opportunities for market-driven innovations. The standards would ensure that all components are Interoperable. The platform components would be designed to be loosely coupled to reduce vendor dependencies or lock in.

2.9. Virtualized Infrastructure Distributed Utilization

IFMS 3.0 Platform will have a common Infrastructure layer catering to the compute, storage, network, and operating system requirements of all common platforms/ applications and all business applications. The infrastructure will be enabled as a Service model and virtualized to the extent possible. Through service-orientation and virtualization for resource provisioning, infrastructure can be deployed and managed dynamically and with minimal impact on the business by reducing the expected downtime. This approach ensures that the resources are monitored effectively and adjusted for optimal performance and efficiency.

2.10. Performance, Scalability & Security by Design

All components of the IFMS 3.0 platform shall be designed for the current and envisioned scale and volume of operations of all lines of business. The design of all technology capabilities needs to ensure high performance and availability, ability to scale horizontally as well as vertically, and an end-to-end security of services and information. The design would consider the evolution of the system, for enabling change and expansion as and when required.

2.11. Proactive Monitoring & Response

With technology enabling end to end automation and the rising customer expectations on the availability and quality of digital services, it is most essential to have continuous monitoring of the performance of the services in terms of business and technology KPIs to be monitored. IFMS 3.0 would enable end to end monitoring of services, applications, data, and infrastructure, to enable pro-active alerts on any issues that occur or are suspected to occur and enable IFMS to take timely action to respond to these alerts. This monitoring capability would also be integrated with the analytics solution of IFMS to further strengthen the capability.

3. Assumption and Dependencies

below points are taken as assumptions and dependencies for the project:

1. Raj eVault is an enterprise level document management system where all the documents will be uploaded
2. Raj Sewadwaar is an enterprise grade ESB solution which will be used to expose and manage IFMS 3.0 APIs and for the integration with any of GoR ecosystem.
3. IFMS 3.0 will be setup in Rajasthan State Data Centre which is managed by IT department
4. RSDC manages the infrastructure and network monitoring and security and deploys the necessary tools.
5. Raj SSO will be used for user authentication and the same token should be used further to get the user Authorization token.

4. Target Architecture

Going forward, the most fundamental aspect for a technology transformation for IFMS operations would be to design and enable a framework which will be the foundation of IFMS journey as a future ready services agency. The application landscape should be 'modular' in nature allowing IFMS to change/ modify any of the applications without necessarily requiring significant work on others. For the IFMS Operations to be technologically competitive, The solution also needs provision for seamless interaction with third party services, including with services of Banks, vendors etc to fulfil the business needs and aspirations of IFMS.

IFMS operations can be broadly segregated into four elements:

Core Solution Components

These are core functions of the IFMS operations which would be integrated with various external GoR and third-party systems:

- Budget
- Expenditure
- Revenue
- Accounting

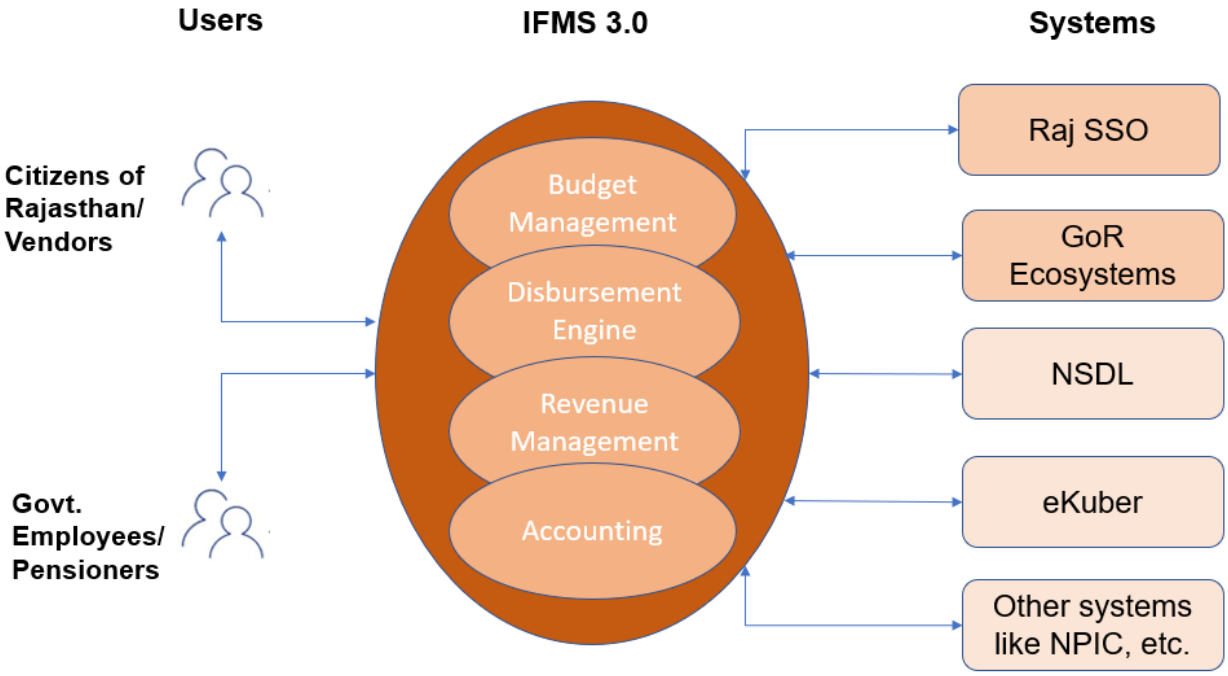
Common Solutions

These solutions are essentially support functions which will enable IFMS to run more efficiently. These are

- BI - Reports and Analytics
- Rajasthan State IDMS / Local IDMS (RajSSO)
- Content Server/Document Management System (Raj e-Vault)
- GoR Ecosystems like RPP, Rajkaj, etc.

Integrated Solutions

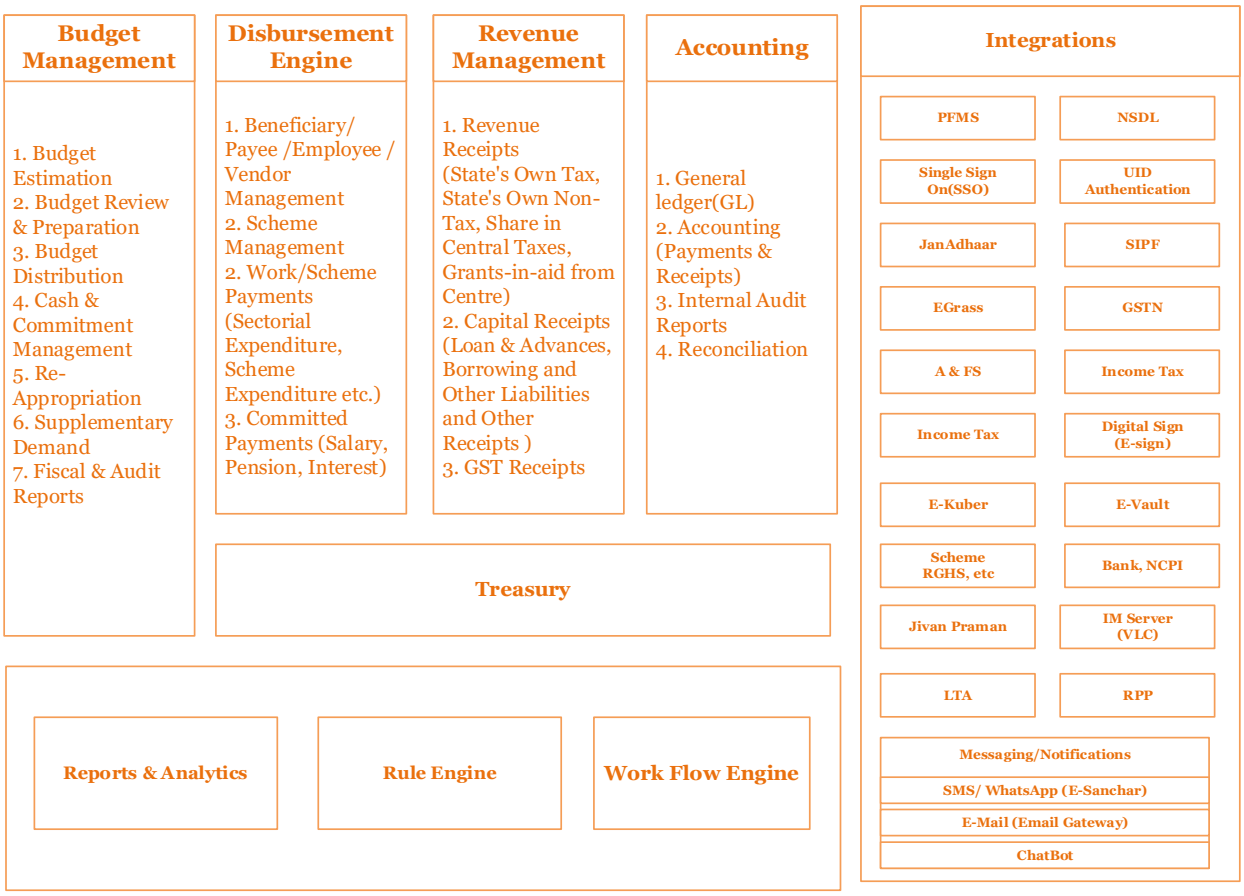
These would be essentially third-party owned solutions with which the IFMS Operations technical stack will be interacting closely for various business functions and processes. For example, SNA needs an integration with Banks etc.



IFMS 3.0 Solution Context

4.1. IFMS Functional Architecture

From a functional standpoint, the IFMS Operations will have two aspects – a) core applications of IFMS operations which will be directly governed by the business policies and processes of the IFMS and b) other related applications and solutions with which these would interact to create a holistic technology solution. It is depicted in the following diagram:



IFMS 3.0 Functional view

Functional Architecture block wise description

#	Functional Block	Description
1	**Budget**	
1.1	Budget Planning and Preparation	The Budget Planning and Preparation module of the IFMS is envisaged to support the Finance Department, line Departments and other key stakeholders across all the stages of the budget planning and preparation.
1.2	Budget Allocation	This module will also support in recording and approving the requests for virements (re-distribution, re-appropriations), and supplementary demands and facilitate surrender of savings.
2	**Expenditure**	
2.1	Administrative Approvals, Technical and Financial Sanctions	The Administrative Approvals, Technical and Financial Sanctions module will support in the accordance of administrative and financial sanctions for various expenditure categories. The administrative sanctions will be validated and approved against the distributed budget for the concerned entity. This module will also be

#	Functional Block	Description
		interfaced/integrated with the e-procurement module to enable a two-way information exchange. Information on accorded sanctions will be shared with the e-procurement system for the purposes of conducting procurement and the commercial terms of the successful bidder will be shared with the IFMS 3.0 for the purposes of creating and approving requests for financial sanctions. This module will also be linked with the cash management module to validate the quantum of financial sanctions issued month/quarter wise against the utilization ceilings. The following lists the key features of the sanctions module: • Project Profile Creation for proposed works/ non-works • Technical Sanctions for works • Administrative Approvals for works/ non-works • Integration with e-Procurement • Financial Sanctions • Sanction recording for on-going projects • General Requirements • Project Progress Tracking • Reporting Requirements (incremental based on added functionality)
2.2	Cash Planning and Management	The Cash Management module will support in submission and finalizing the monthly/quarterly cash flow forecasts and borrowing requirements based on the cash flow analysis. It will also support in updating the monthly/quarterly cash forecast based on the actual progress of revenue collections, expenditure incurred and financial sanction requests. This module will be linked with the sanctions management and the budget allocation and distribution modules for updating the cash forecast and for approving any requests for financial sanctions. The following lists the key features of the cash planning and management module: • Cash flow forecasts for Receipt Estimates • Cash flow forecasts for Expenditure Estimates – Salary, Wages & Pensions • Cash flow forecasts for Expenditure Estimates – Establishment Expenditure for which budget is prepared centrally

#	Functional Block	Description
		• Cash flow forecasts for Expenditure Estimates – Centrally Sponsored/ Central Sector Schemes • Cash flow forecasts for Expenditure Estimates – Other committed expenditure • Cash flow forecasts for Expenditure Estimates – Debt estimates • Submitting of Cash flow forecasts by Line Departments – for project/ schemes expenditure • Finalizing Cash flow forecast statements • Determining borrowing requirements • Reporting Requirements
2.3	Bill Creation, Expenditure Processing and Reporting	The Disbursement Engine module of the IFMS 3.0 will support the DDOs in the online preparation and submission of bills. This module will facilitate in recording the various deductions and creation of journal entries for the same. It will also support in maintaining the contracts and vendors databases, which will facilitate in the bill preparation process. This module will also be integrated with the e-Kuber system of the RBI for the payment processing and support in establishing a cyber-treasury function for centralized payment processing. This module will also support in creating refund bills against challans and will interface with the e-GRAS system / receipts management module for the exchange of refund data. The following lists the key features of the bill creation and espenditure processing and reporting modules: • User management • Create new bill- Employee Payment Related • Create new bill- Others • Special provision for AC/ DC Bills • Utilization Certificates • Advance bills • Advances/ Deposits • View pending bills (for action) • View bill history/ status • Payments processing- General • Payment Processing– Ways and Means management • Vendor Management • GIA Moinitoring • Interface with e-Procurement System

#	Functional Block	Description
		• Interface with PFMS • Reconciliation with RBI • DBT Payments • Out of Treasury Disbursements • Reporting • Bill Processing SLAs • Master data sets
2.4	Debt Management	The Debt Management module will support in recording the details of the various debt instruments availed by the State, which include the receipts and debt-servicing schedule for institutional loans and open market borrowings. It will also support in recording the loans and advances, guarantees and investments made by the State Government towards the various public sector enterprises and other corporations. This module will be linked with the expenditure management module for creating and processing debt-servicing payments. It will also be linked with the budget preparation module to support in preparation of the debt receipts and disbursements estimates based on the recorded schedules. The following lists the features of the debt management module: • Recording debt servicing requirements/ repayment schedule for institutional loans • Recording debt servicing requirements/ repayment schedule for Market Borrowings • Guarantee and Investments Monitoring • Debt Repayment– Ways and Means management • Government Loans and Advances
2.5	Employee and Payroll	The employee database will support in maintaining the service records of the employees of the Rajasthan government of different cadres. It will support in managing the cadre strength as well as in maintaining records of all service related aspects such as transfers, promotions, postings, etc. for the government employees. This module will also be linked to the pensions module for generating alerts to initiate pension files based on the expected retirement dates as well as providing the digital service records required for pension file preparation and processing.

#	Functional Block	Description
		The Payroll Processing will support in maintaining the payroll database for all State Government employees and other contractual/wage employees for processing the monthly payroll. It will support in auto-generation of the paybills based on the monthly variable information to be submitted by the DDOs in the State. It will support in auto-computation of the various salary related deductions and in creating journal entries to record the same. This module will be integrated with the e-Kuber system to facilitate centralized payroll processing based on the payroll details. The following lists the features of the employee and payroll module: • Employee Information Management • Employee Self-Services • Interface with NSDL/ RBI • Configuration flexibility in system- Organization structure, etc. • Master Data sets
2.6	Pensions Management	The Pensions Management module of the IFMS 3.0 will support in creation and processing of the pension files. It will be linked with the employee database to identify employees due for retirement and initiate the pension file preparation based on the same. The pensions module will also be linked with e-Kuber system of RBI for processing the pension disbursements. The following lists the features of the pensions module: • Pension Proposal File creation • Employee interface on system • Provisional pension and gratuities • Department interface on system • Pension Processing at Directorate of Pensions • Pension payment from Treasury • Pensions Portal • Pensioner's Page • Life Certificate Certifier • Commutation against pension • Grievance Redressal • Reports • Administration

#	Functional Block	Description
		• Master Data sets
3	**Revenue**	
3.1	Receipt Management	The Receipts Management module of the IFMS will serve as a comprehensive receipts collection portal, which will allow both taxpayers, citizens and DDOs to create challans and deposit receipts to the consolidated fund of the State, as well as to various deposit accounts. It will support in processing both online payments and will be linked to the various agency bank payment gateways as well as the IT systems of the various revenue earning departments. It will also support in providing real-time information on the revenue collections of the State and in preparation of the government receipts accounts
4	**Accounting**	.
4.1	GL, Accounting, Reconciliation	The Accounting and Reconciliation module of the IFMS 3.0 will comprise the General Ledger and the various sub-ledgers for expenditure and revenue collections. It will be linked with all the other modules of the IFMS 3.0 to support in creation of ledger entries for various transactions, creating journal entries for transfer entries and suspense payments. It will support in maintaining the accounts information for the consolidated fund, public accounts and contingency fund. This module will also facilitate in generating real time reports on the government accounts for management reporting. The following lists the key features of the accounting and reconciliation module: • Defining and configuring accounting rules for various types of transactions • Accounting for public accounts and deposits including contingency fund • Managing public account heads • Automatically updating the general ledger • Create transfer entries for the deductions • Create suspense entries when failed payments are recognized • Create journal entry for the suspense entry post reconciliation • Create balancing entries for transactions as per the defined accounting rules • Integration with the VLC system of the AG Supporting cash-based accounting, able to migrate to a hybrid accounting system and support for future migration to full accrual-based accounting

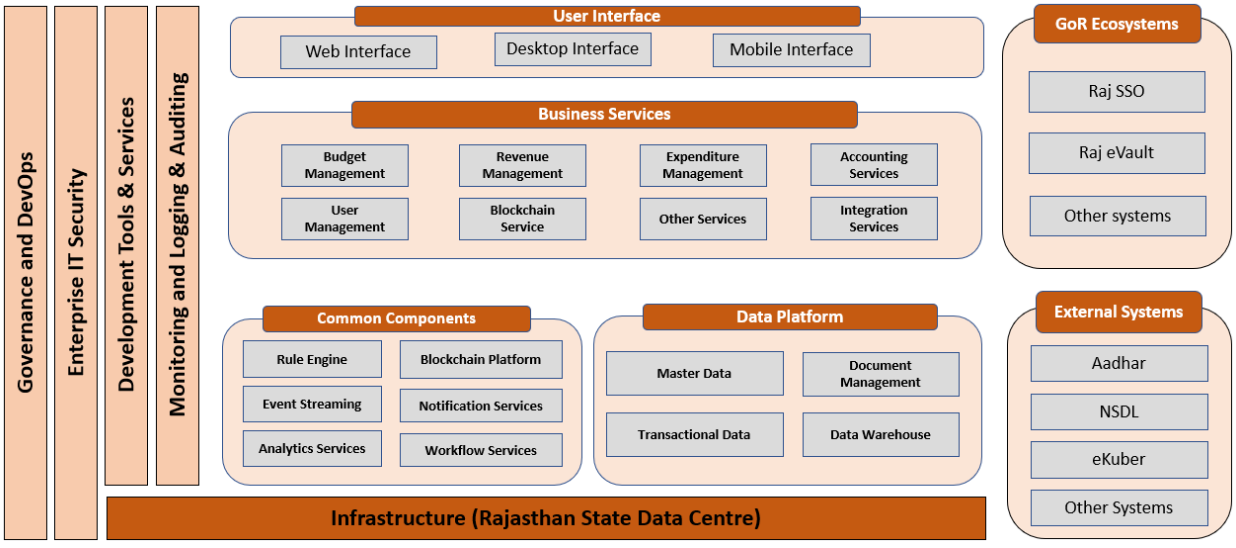
#	Functional Block	Description
4.3	Internal Audit	Audit module will support to record, track objections recorded during analysis and scrutiny of government transactions, and actions taken by the concerned departments/ agencies against each of the objections recorded.
5	**Notifications**	
5.1	Alerts and Communication	The stakeholder will receive timely alerts at each stage of operations with relevant details applicable. A communication (SMS, Email) will be sent to the user in case of a relevant event.
5.2		
7	**Reports & Analytics**	TBU
7.1	Volumetric Analysis	TBU
7.2	Performance and Utilization	Reports based on KPI's will be generated along with utilization of various resources region wise/circle wise. Reports can be drilled down to individual counter level. These will help to better utilize and plan various resources.
7.3	Stakeholder Analysis	Reports will be generated for various stakeholders with volumes and KPI benchmarks, the reports will help with management of stakeholders and give a holistic view of the engagement.
7.4	Predictive Analysis	Reports will be generated using past data to predict the potential impacts. The analyses will give IFMS an objective view of the risks and rewards involved in each potential decision and allow them to plan better for the future.
7.5	Visual Dashboard	Visual dashboards will give stakeholders a real-time overview of operations in the form of graphs, charts, summaries and other information reports.
12	**Enterprise Shared layer**	This layer is required to easily integrate, monitor applications for IFMS applications and internal/external communication. This layer contains common frameworks, Infrastructure, Governance, Monitoring and Security which will be utilized to support the core IFMS solution
12.1	Mobility	Mobility solutions will be a major part for IFMS Solution. Multiple sets of Mobile applications are required: End User Application Mobile application for Internal Users
12.2	Integration	The IFMS Solution will need to Integrate with Various external (Payment Gateways, 3rd party vendors, other government organizations) and internal Applications like Banking, RBI AG
12.3	Infrastructure	To run effectively the solution will require infrastructure like network, computers and peripheral devices like weigh scale and printers

#	Functional Block	Description
12.4	IT Governance	The IFMS needs to cater to a lot of stakeholders. To cater to all the stake holders, it will require standardized IT policies and procedures. GRC framework and change management
12.5	IT Monitoring	As IFMS solution has many components and each component needs to work independently as well an integrated solution, IT monitoring is a must. To seamlessly run the solution, we will need application monitoring, database monitoring, integration monitoring, infra and SLA monitoring
12.6	Security	Due to large number of integration both internal and external IT security is must. The solution needs to ensure appropriate levels of confidentiality, integrity and availability in conjunction with applicable regulation and legislation, so an end-to-end security solution for data and services is needed.

TABLE 1 TO BE FUNCTIONAL OVERVIEW

4.2. IFMS 3.0 Technical Architecture Framework

IFMS 3.0 platform will be built using a structured architecture framework. The framework identifies the first level of building blocks as architecture layers to standardize the landscape to the extent possible by building common/shared layers that shall be used (re-used) by all business-specific solutions, thus ensuring a strong control on redundancy. The layered framework is shown below and each of its layers, their purpose, and the key capabilities/ building blocks of each layer is also described below.



IFMS 3.0 Enterprise view

The framework ensures that complexity and redundancy are controlled and governed across the landscape., while enterprise standard/ shared solutions will be put in place to be used across the core solutions enabling the digital service delivery across all IFMS lines of business.

The framework identifies the layers that are common requirements across any digital initiative/ solution realization in today's time to build a high performing, governed, and monitored IT landscape. All solutions for enabling security, integrations (internal as well as external), enterprise data management, infrastructure management, connectivity/ network management mobility management, access channels enablement, reporting, business intelligence, analytics, IT monitoring, IT Governance, and IT operations – shall be instantiated as shared solutions across IFMS. In addition to these certain applications shall also be instantiated as shared/ re-usable applications across the enterprise. Using these shared building blocks core solutions for the four lines of business i.e., Budget, Expenditure, Revenue and Accounting will enable their core solutions, in principle without duplicating these solutions; unless there is a product dependency or any specific requirement which has to be approved as an exception after ensuring integration is enabled with the enterprise shared solution.

4.3. Components of the Framework / IFMS 3.0 Landscape

Below mentioned Sections briefly describes the layers and its components.

User Interface

IFMS 3.0 platform shall offer access to its services through the following channels and deliver an omnichannel experience to users:

1. Web/ m-browser – IFMS 3.0 platform shall enable information dissemination and access to all services through a responsive web interface accessible over the internet on web browser through any device.
2. Mobile Apps – IFMS 3.0 Platform shall also provide mobile apps for various stakeholders, enabling omnichannel access to services and information.
3. SMS - SMS services shall be used for multi factor authentication (MFA) (e.g., OTP) authentication, sending alerts / intimations / status updates etc. to users, and to enable users to query status of any of their service transaction
4. Email – IFMS 3.0 shall enable a paperless transaction ecosystem. Email services shall be enabled to send automated receipts/ invoices/ alerts / intimations etc. to registered email ids, based on preferences of individual users.
5. WhatsApp - IFMS 3.0 shall enable a paperless transaction ecosystem. WhatsApp services shall be enabled to send automated receipts/ invoices/ alerts / intimations etc. to registered mobile numbers, based on preferences of individual users.

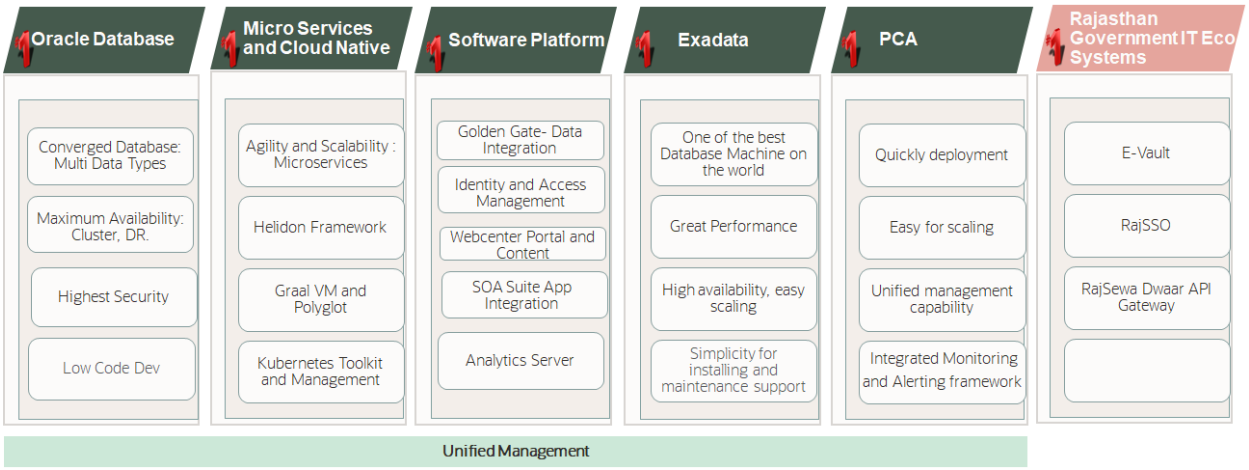
IFMS 3.0 platform would be built using endpoint device-agnostic, where-in the following components would be an integral part of the endpoint infrastructure.

1. End Computing Devices - To access all the relevant applications and network services, the IFMS 3.0 platform will be provisioned to connect various end computing devices such as desktop, laptop, and mobile, etc. IFMS 3.0 platform will be provisioned for BYOD as well as and when required by IFMS.

2. Printers/ Scanners – Printing and scanning are integral part of the services offered by IFMS. Devices required to support and enable these requirements shall be provisioned at IFMS service delivery locations.

Infrastructure Layer

IFMS3.0 framework will utilize best of breed solution components from Oracle technology stack along with Raj Government solution components.



Below mentioned Sections briefly describes the layers and its components.

Hardware

Exa Data: Exadata is a combined hardware and software platform that includes scale-out Intel x86-64 compute and storage servers, RoCE networking, persistent memory (PMEM), NVMe flash, and specialized software. Oracle Exadata allows enterprises to run any Oracle Database workload with the highest performance, scale, availability, and security on fully compatible cloud and on-premises infrastructure. Exadata uses a scale-out design with unique optimizations that include persistent memory, SQL query offload, and built-in resource management to optimize performance for OLTP, analytics, machine learning, and mixed workloads running in consolidated environments.

Private Cloud Appliance (PCA): Preconfigured with servers, storage, and management software reduce deployment and management complexity. Oracle Private Cloud Appliance is an integrated infrastructure system engineered to enable rapid deployment of converged compute, network, and storage technologies for hosting applications or workloads on a guest OS. It is a data center-class system that provides incremental and scalable performance optimized for consolidation of mixed workloads

Storage

Oracle ZFS Storage Appliance is a high performance, enterprise storage system that is optimized for Oracle workloads and cloud integration. It is a unified storage system that allows customers to consolidate file, block, and object storage on a single platform. Oracle ZFS combines high all-flash performance with petabytes of storage capacity, allowing customers to run all workloads at peak speed. Built-in Oracle Database integrations, including automated prioritization of database IOs, enable IT departments to optimize workload performance and reduce administration to achieve low total cost of ownership (TCO)

Platform

Oracle VM Manager, Oracle Database, Oracle Cloud Native Environment/Modules, SOA, WCC, OAS

Backup Infra

Zero Data Loss Recovery Appliance (ZDLRA)

Oracle's Zero Data Loss Recovery Appliance (Recovery Appliance or RA) is a ground-breaking data protection solution that tightly integrates with the Oracle Database. It eliminates data loss and dramatically reduces data protection overhead on production servers. In addition, the Recovery Appliance continually validates the integrity and recoverability of the data, scales to protect thousands of databases, and protects backups across the full lifecycle, including disk backup, cloud archiving, remote replication and tape archiving

Aligning to the architecture principle of Virtualized Infrastructure Distributed Utilization, the IFMS 3.0 platform will have a common infrastructure layer catering to the network, compute, storage, and operating system requirement of all common solutions as well as applications and for all core business applications.

To address the infrastructure requirement of IFMS 3.0, the infrastructure architecture would be based on service-oriented network architecture (SONA) in which functionalities can be added to the infrastructure as and when required. IFMS 3.0 infrastructure layer would offer the following for enabling digital service delivery across IFMS:

1. Integrated Transport:
2. Integrated Services:
3. Integrated Application:
4. Data Center (RSDC): IFMS 3.0 platform would be built using common use of applications across the line of business as a shared enterprise standard application. To host these applications, it is recommended to use a Service-centric data center where-in IFMS can pool compute, storage, and resources that are provisioned to support applications over the data center network by using Hyper-Converged Infrastructure. Because the network touches and can control all the components, the network can be used to integrate all the applications and services; network technology actively participates in the delivery of applications to end-users.
5. Deployment Management:
6. Business Continuity solution:
7. DRM/SRM leverages the benefits of virtualization and can also take advantage of the Software-Defined Data Center (SDDC) architecture, integrating with other solutions, such as SDN (network virtualization) and hyper-converged infrastructure.

Business Services

As a core Architecture principle IFMS 3.0 platform is planned to be built on Microservices based architecture to the extent possible. There would be multiple microservices in the IFMs ecosystem and these would need to communicate with each other. For interservice communication Istio based service mesh would be leveraged. This component provides capabilities to control and trace the interservice communication.
There would be following modules, under each of these modules there would be multiple microservice with various endpoints/ services developed for a specific purpose.

- **Expenditure Management:** Microservices for employee management, expenditure management, pension management and other function/process related to expenditure.
- **Budget Management:** Microservices for managing budget preparation, allocation, commitment management, approvals and other functions related to Budget.
- **Revenue Management:** Microservices for receipt management, revenue tracking and other functions related to Revenue.
- **Accounting Services:** Microservices for managing general ledger/ accounting, internal audit, and other Accounting related functions.
- **User Management:** Microservices for user management, login, role management and other user related functions.
- **Blockchain Services:** Microservices to interact with Blockchain platform
- **Integration services: Microservices for integration with GoR ecosystem and other external systems.**
  Other Services: Any other microservices which will not be covered under the above-mentioned categories.

Enterprise IT Security

IFMS Services will ensure appropriate levels of confidentiality, integrity, and availability in conjunction with applicable regulation and legislation. To achieve this, IFMS adopt or enhance the following security capabilities for ensuring an end-to-end security of the data and services. These security capabilities will be common across the complete IFMS 3.0 technology landscape and shall be across all other solutions:

1. **Identity & Access Management** – Security solutions to define policies and enforce implementation of identity & role management and access authentication & authorization. These will comprise a range of technologies to manage user authentication, authorization, and accounting. The primary objective of such technologies would be to maintain user identity, roles, and permissions within the IFMS IT ecosystem. The identity and access management layer will also be responsible for managing logging mechanisms, authentication mechanisms, and policies for managing privileged users such as system, application, and database administrators.
2. **Digital Signatures** – The digital signature solution will enable stakeholders/systems to digitally sign messages/ documents for ensuring the integrity & authenticity of the content generated/ exchanged on the IFMS 3.0 platform. Digital signature certificates will primarily be enabled for approvers and officials responsible for generating signed documents.
3. **Application Security** – Solution for developing, adding, and testing security features within applications to prevent security vulnerabilities against threats such as unauthorized access and modification. Security controls would be built for applications from all access channels like web, mobile, etc. A secure SDLC framework leveraging DevOps will be used for any application development activities. Having a DevOps approach will ensure that security is built-in during the development of applications. Continuous threat and vulnerability

management have to be adopted to identify, track, and resolve vulnerabilities and exploits within the IT landscape of IFMS.

4. **Data Security** – Solution for protecting data from unauthorized access and data corruption throughout its lifecycle. Data security includes data encryption, hashing, tokenization, key management, and data loss prevention practices that protect data across all applications and platforms. Data security will also cover database activity monitoring to monitor and manage database activity including the activity of privileged users, database modifications, protection from database-specific threats such as SQL injection attacks, and log management.
5. **Network & Perimeter Security** – IFMS system will be deployed in Rajasthan State Data Center for network security to prevent any unauthorized access to IFMS's resources, and provisioning of Security devices like web, email and antivirus gateways, anti-phishing and anti-spam, IPS, Anti-APT, Anti-DDoS, Web Application Firewall, and Network Firewalls help protect the IFMS network against malicious attacks. RSDC Team is responsible for maintaining the Network security.
6. **Infrastructure Security (Host & End Point)** – Solution for implementation of a layered defense approach for security of IFMS's IT infrastructure using multiple mitigating security controls to protect IFMS's resources and data and ensuring all data communication with IFMS's infrastructure locations (DC/ DR/ Cloud) to be governed by IFMS's Information Security Policy. This includes endpoint security solutions such as antivirus, host intrusion prevention systems, endpoint DLP solutions, and endpoint detection and response (EDR). These tools provide comprehensive malware protection, data leak protection, active intelligence on real-time threats, and protection from advanced endpoint attacks.
7. **Audit & Compliance** – Solution for performing periodic audits (both external and internal) to inspect the functioning of all IFMS 3.0 platform components, comprising of a security audit, quality checks/ compliance, and management of audit compliance SLAs. Vulnerability scanning tools will be deployed to run infrastructure and application scans to proactively monitor vulnerabilities.
8. **API Security**: IFMS system will be built on Microservice Architecture which makes it vital to make all these microservices secured. Inter microservice communication will be through a service mesh (Istio) which enables secured and traceable communication across services. Additionally, mutual TLS between services can be configured. RajSSO based Authentication & Role based Authorization will be enabled to access control the services. Service will not be exposed directly to the outside IFMS ecosystem, there would be a Gateway (istio gateway) to expose the services and further it will be safeguarded by Raj Sewa Dwaar for exposing to the external worlds (outside GoR ecosystem).

Integrated IT Monitoring Layer

The IFMS 3.0 platform is intended to be built as a performance-oriented IT landscape enabling seamless service delivery with minimal disruption and immediate response to any threat or incident. Extending the principle of "Proactive Monitoring & Response" IFMS 3.0 platform will offer the required tools and technologies to enable the following aspects of end-to-end monitoring of the IFMS 3.0 IT platform:

1. **Application Monitoring** –APM module will be used for the complete application performance monitoring (APM) and visibility into the health of key applications, their supporting infrastructure, and knowledge of issues that can impact the availability of key systems and end-users experience as well.
   Grafana Dashboard with Prometheus Data Source can be used to monitor the Application/ microservices. Prometheus is an open source monitoring system for which Grafana provides

out-of-the-box support. Prometheus pulls (scrapes) metrics from a client (target) over http and places the data into its local time series database that you can query using its own DSL. Prometheus uses *exporters* that are installed and configured on the clients in order to convert and expose their metrics in a Prometheus format.

Grafana is open-source platform for monitoring and observability. It allows you to query, visualize, alert on and understand your metrics no matter where they are stored. Create, explore, and share dashboards with your team and foster a data-driven culture.

2. **Database Monitoring** - Database Performance Analyzer (DPA) will be used to improve database response time. It will help to quickly identify exactly which steps in which operations may be causing application delays, making it easier to reduce response time and improve the end-user experience as well. AWR Report and OEM Monitoring can be used for monitoring
3. **Network Monitoring-** For network uptime and least response time of network services, network monitoring tools will be used by RSDC team in IFMS 3.0.
4. **Infrastructure Monitoring -** To make sure the health of underlying infrastructure, infra monitoring tools will be deployed and used by the RSDC team to make sure the availability of the underlying infrastructure.
5. **Integrations Monitoring** – IFMS 3.0 is envisaged to be an integrated ecosystem, therefore, to ensure the success of the platform it is essential to monitor availability and response times of all integration touchpoints. Tools shall be deployed and configured by the respective GoR ecosystem teams to monitor all integrations in the IFMS 3.0 landscape.
6. **Security Monitoring –** Monitoring the entire IFMS ecosystem with respect to security will be setup using different tools, which would monitor following parameter. Ecosystem level security will be monitored by the RSDC team and application level security monitoring will be configured in the CI/CD pipeline.
    - Network security
    - Infrastructure Security
    - Server Security patching
    - Server Antivirus updates
    - Server open port scanning
    - Code and application-level vulnerability

Integrations - Government of Rajasthan (GoR) Ecosystems

As a core principle, the IFMS 3.0 technology landscape will be built on the concept of an integrated/unified platform of systems. All integrations and information exchange with and of the GoR systems shall also be enabled using APIs. This component shall offer all capabilities required for managing consumers of APIs exposed by IFMS and producers of APIs that are exposed by external systems for IFMS using Rajsevadwar. Following are the list of GoR Ecosystems to be integrated with IFMS 3.0. Monitoring of APIs being exposed and consumed is provided by Rajsevadwar.

- A&FS (Works)
- SIPF
- Janadhaar
- Scheme payments (respective depts.)
- PFMS
- Jeevan Pranam
- eSign

- eVault
- eSanchaar
- RPP
- IM Server
- eGRAS
- RajSSO
- LTA
- RGHS

Integrations - External Systems

While the aim of the IFMS 3.0 platform is to move towards an almost real-time information exchange, the nature of the businesses that IFMS operates rely heavily on scheduled data exchanges with external stakeholders. Such exchanges are already automated and integrated with backend systems. IFMS 3.0 shall continue to provide capabilities to support message management using Oracle SOA suite which also provide the capability of monitoring and metering of the APIs. Following are the list of external systems to be integrated with IFMS 3.0

- eKuber
- NSDL
- NPCI
- UID
- GSTN
- Income Tax
- PFMS

4.4. Proposed Technical Architecture

IFMS is standardizing its technology architecture at the enterprise level, consolidating the technology landscape for all lines of businesses. To enable the functional requirements of the IFMS operations, there is a necessity of a unified enterprise architecture encapsulating different solutions and applications for varied business services and purpose. It is envisaged that the proposed enterprise architecture will provide the business functions with enough flexibility to introduce new technology solutions for business processes and functions, as well as to modify or enhance the existing solutions, without the need of disrupting the larger technology landscape. The technology architecture, therefore, will be predicated on this ‘modular’ approach, aligned with the IFMS 3.0 architecture principles of maximizing re-use.

While the technology solutions and modules of the IFMS operations will be closely inter-twined, interactions with solutions of other business functions and common applications are necessity to be addressed by an enabling layer for API based access. For instance, the revenue and expenses of the IFMS will need interlinkages with both the Banking functions (for digital payments and clearance) as well with Finance and Accounting solutions for costing and budgeting.

As a principle, IFMS operations will be leveraging the common IT technology and infrastructure layers like Enterprise IT Security, Enterprise Mobility and Access Layer, amongst others, encapsulated by the Enterprise Integration Layer, for creating a holistic technology framework. The schematic view of such a unified technology architecture needed for fulfilling the business objectives of IFMS operations is provided below.

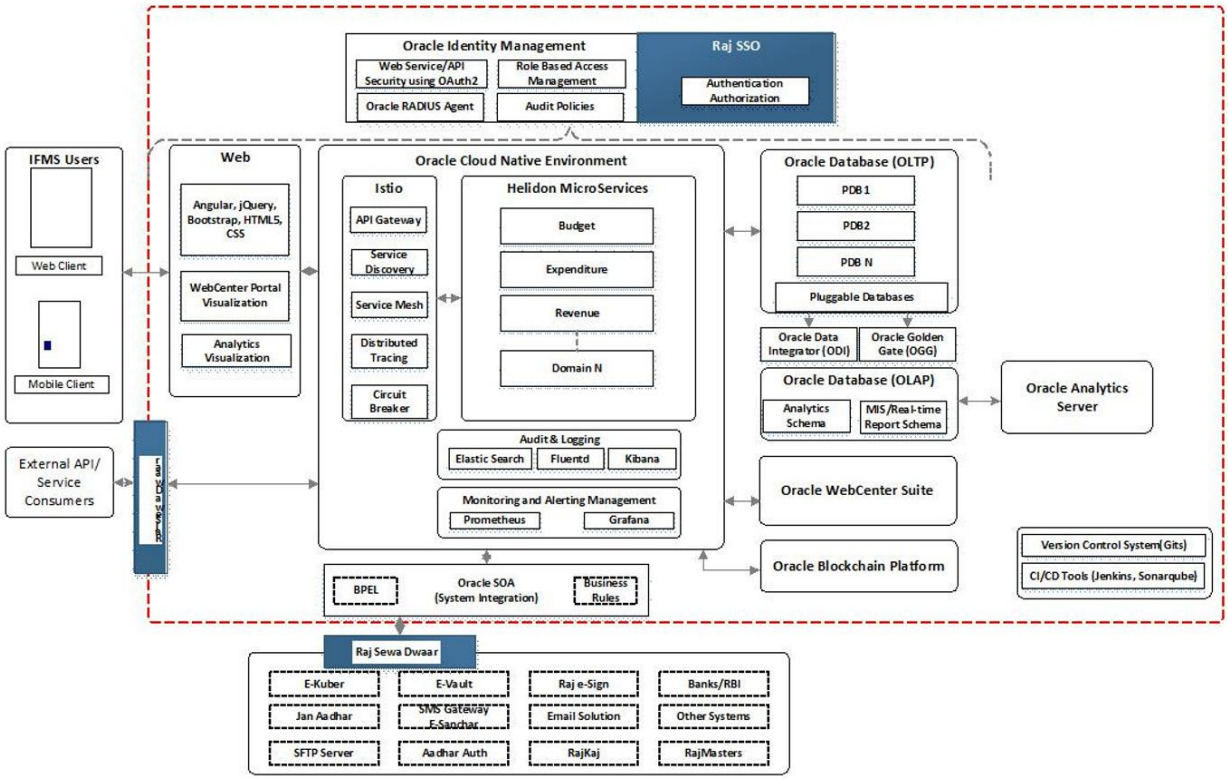


FIGURE 1 SCHEMATIC VIEW OF THE TECHNOLOGY ARCHITECTURE FOR IFMS OPERATIONS

The IFMS solution will be built on a structured architecture framework. The framework identifies the first level of building blocks as architecture layers to standardize the landscape to the extent possible by building common/ shared layers that shall be used (re-used) by IFMS solutions, thus ensuring a strong control on redundancy.

Technical Architecture Components

Web Components

IFMS 3.0 platform shall offer access to its services through multiple channels like web browser, mobile application and will be a combination of custom application development using Angular, JQuery, React native and Portal development framework Oracle WebCenter Portal. The custom web application should be highly scalable and fault tolerant to handle the high traffic of the state users, for this purpose the UI component of IFMS 3.0 can also be developed as micro UIs and deployed as a containerized component.

Oracle Cloud Native Environment

Oracle Cloud Native Environment is a fully integrated suite for the development and management of cloud-native applications. Oracle Cloud Native Environment is a curated set of open source projects that are based on open standards, specifications and APIs defined by the Open Container Initiative (OCI) and Cloud Native Computing Foundation (CNCF) that can be easily deployed, have been tested for interoperability and for which enterprise-grade support is offered. Oracle Cloud Native Environment delivers a simplified framework for installations, updates, upgrades and configuration of key features for orchestrating microservices

Micro Service Framework – Helidon

Helidon is a cloud-native, open-source set of Java libraries for writing microservice that run on a fast web core powered by Netty. The framework supports two programming models for writing microservice: Helidon SE and Helidon MP.

While Helidon SE is designed to be a micro framework that supports the reactive programming model, Helidon MP, on the other hand, is an Eclipse Microprofile runtime that allows the Jakarta EE community to run microservices in a portable way.

In both cases, a Helidon microservice is a Java SE application that starts a tinny HTTP server from the main method.

Istio

Istio is a service mesh—a modernized service networking layer it extends Kubernetes to establish a programmable, application-aware network using the powerful Envoy service proxy. It is a popular solution for managing the different microservices that make up a cloud-native application. Istio service mesh also supports how those microservices communicate and share data with one another.

Elasticsearch, Fluentd and Kibana (EFK) Logging Stack

Elasticsearch is a real-time, distributed, and scalable search engine which allows for full-text and structured search, as well as analytics. It is commonly used to index and search through large volumes of log data but can also be used to search many different kinds of documents. Elasticsearch is commonly deployed alongside Kibana, a powerful data visualization frontend and dashboard for Elasticsearch. Kibana allows you to explore your Elasticsearch log data through a web interface, and build dashboards and queries to quickly answer questions and gain insight into your Kubernetes applications.

Fluentd is used to collect, transform, and ship log data to the Elasticsearch backend. Fluentd is a popular open-source data collector

Jaeger is a distributed tracing platform. It can be used for monitoring microservices-based distributed systems:

Distributed context propagation

Distributed transaction monitoring

Root cause analysis

Service dependency analysis

Performance / latency optimization

Oracle SOA

Oracle SOA Suite enables you to transform complex application integrations into agile and reusable service-based applications to shorten the time to market, respond faster to business requirements, and lower costs

Oracle Analytics Server

Oracle Analytics Server (OAS) is an on-premises self-service visualization and augmented AI analytics platform. Built on a proven and modern technological foundation, it supports complex workloads while providing timely insights to users across an enterprise. Organizations can now modernize their analytics platform by providing easy-to-use interfaces for all users who need to access curated data, self-serve by importing or blending data, perform analysis, or distribute reports securely via mobile, tablet, and all modern browsers.

Oracle Identity and Access Management

Identity and access management (IAM) manages the end-to-end lifecycle of user identities and entitlements across all enterprise resources. It is a foundational control of security as it authenticates users and regulates access to systems, networks and data.

Oracle Database

OLTP

OLTP or Online Transaction Processing is a type of data processing that consists of executing a number of transactions occurring concurrently. These transactions traditionally are referred to as economic or financial transactions, recorded and secured so that an enterprise can access the information anytime for accounting or reporting purposes.

OLAP

Online analytical processing (OLAP) is a technology that organizes large business databases and supports complex analysis. It can be used to perform complex analytical queries without negatively affecting transactional systems.

Oracle Web Center Content

Oracle WebCenter Content manages information you use every day. Information found in e-mails, reports, document, memos, slide presentations, and more. Oracle WebCenter Content stores it, organizes it, and secures it so that only the people needing the information have access to it. And most importantly, it helps you find it quickly when you need it and view it through a standard web browser, even if you do not have the software that created the document.

Raj Government Systems

IFMS 3.0 will be integrated with various GoR ecosystems. Different application serves different business need and use of the centralised ESB solution (Raj Sewa Dwaar), State's SSO solution (RajSSO) and the centralized document management system (Raj eVault) is mandated by design to be used in IFMS 3.0.

Raj Seva Dwaar:

Rajasthan State Government has an ESB solution called ‘Raj Sewa Dwaar’ where all the GoR ecosystems services are exposed. IFMS 3.0 will be exposing its services through Raj Sewa Dwaar and integrating with all the required GoR ecosystems through this only.

Raj SSO:

Raj SSO is the centralized platform where user authentication is managed. IFMS 3.0 should be accessed through Raj SSO.

Raj eVault:

All the documents getting uploaded in any of the process of IFMS 3.0 should be uploaded in Raj eVault which can further be consumed by any other application/ system in GoR.

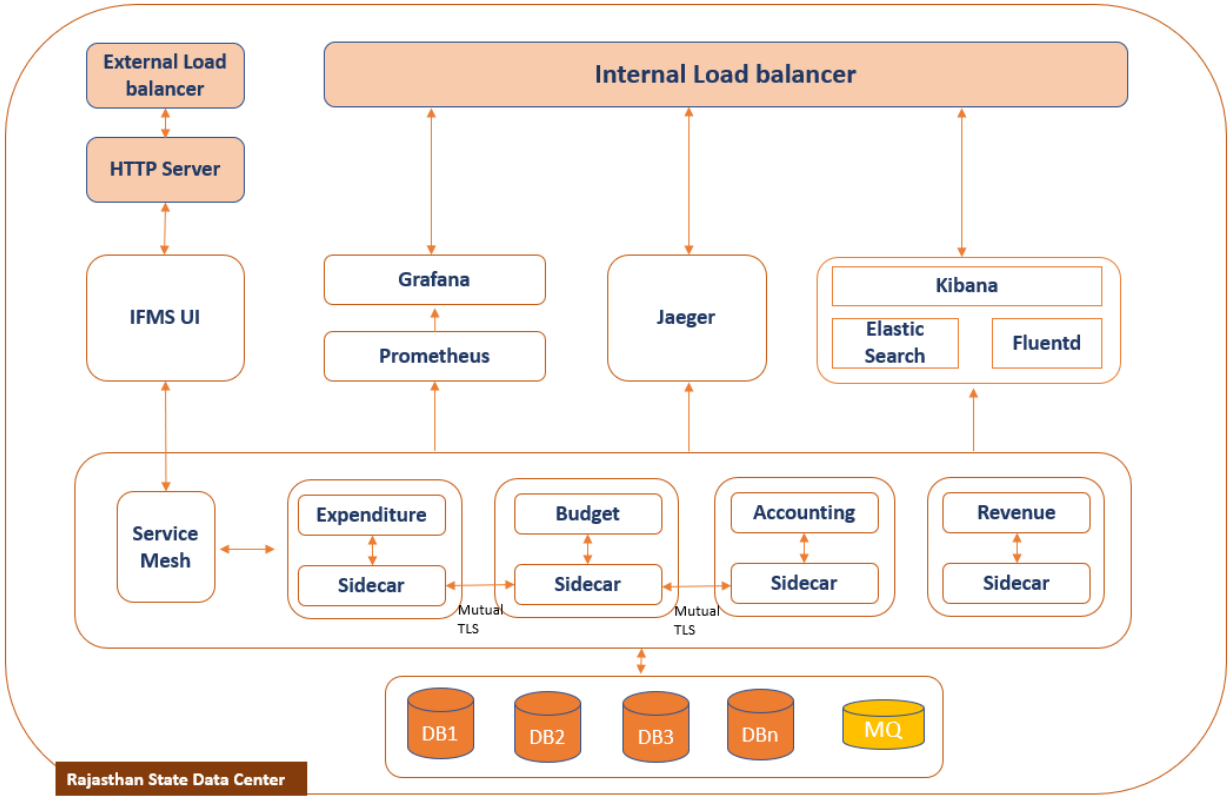
5. Target Architecture - Technical Requirements

5.1. Microservice Architecture and Design requirements

The Target Architecture needs to be designed using Microservices – also known as the microservice architecture – is an architectural style that structures an application as a collection of services that are. Highly maintainable and testable. Loosely coupled. Independently deployable. Organized around business capabilities.

Microservice architecture allows you to maximize deployment velocity and application reliability by helping you move at the speed of the market. Since applications each run in their own containerized environment, applications can be moved anywhere without altering the environment

Microservice Architectural Overview



Here are 8 core components of Microservice Architecture:

1. Microservice [Pod/ Containers]
2. Service Mesh
3. Databases
4. Message Queue
5. Service Discovery
6. Logging
7. Monitoring

Microservice [Pod/ Containers]

Microservices allow applications to be created using a collection of loosely coupled services and are fine-grained and lightweight. Each Microservice is deployed in its own Pod which has one container where the actual business microservices runs and one container (called sidecar or Envoy) which is an integral component of service Mesh. Any incoming/outgoing service communication is done via this sidecar in that particular pod of a specific service. As all the traffic/communication happens via the sidecar its easy to track and manager the interservice communication. There will be one login service [identity provider] which will provide the authorization token based on user-role mapping and this token will be used to authorize the ui-to-service and service-to-service communication for a particular request.

Service Mesh

The Service Mesh is responsible for request routing, composition, and protocol translation. It provides each of the application's clients with a custom API. For most microservices-based applications, implementation of a Service Mesh is very important for a single-entry point into a system. Istio based service mesh helps in interservice communication using the sidecar and also works as a common gateway to expose the microservices outside Kubernetes cluster.

Databases

Microservice owns a private database to capture their data and implement the respective business functionality. Microservices databases are updated through their service API. Theoretically each microservice should have its own separate Database but it brings complexity specially in managing the distributed transactions. Hybrid approach will be followed in IFMS where some of the related microservices may refer the same DB for seamless transactions.

Message Queue

Synchronize and Asynchronized are the 2 types of messages through which they communicate. Every microservice in order to communicate either synchronously or asynchronously with other microservices. "Synchronous – HTTP is a synchronous protocol. The client sends a request and waits for a response from the service. The client code or message sender usually does not wait for a response. To have the asynchronous transactions across microservice, IFMS will be using the Message Queue which will be used by one microservice to push a message/event and read by another microservice for further processing. This additional component in interservice communication makes the two service even more loosely coupled.

Service Discovery

In a microservices application, the set of running service instances changes dynamically. Instances have dynamically assigned network locations. Consequently, for a client to make a

request to service it must use a service-discovery mechanism. A key part of service discovery is the service registry. Istio based Service mesh will be used in IFMS to maintain the service discovery.

Logging

Logging is the key to troubleshoot and manage any application. In microservice there so many individual moving components and each microservice generates its logs locally which makes it necessary to put a mechanism in place which can collect the logs and store at a central storage and provide a traceability of an end-to-end transactions across the microservices. For this purpose, IFMS will be using following industry standards tools and technologies.

Elastic Search – Fluentd – Kibana (EFK): Fluentd is used to capture the logs from each of the microservice container and store it in Elastic Search. Kibana is further used to visualize the logs stored in Elastic Search. With EFK, all the logs from different microservices can be access at a central place.

Jaeger: For a particular business process there would be multiple microservice involved and each microservice has its own input request and outgoing response. In such case to have a single view of end-to-end process is not easy. For this purpose, IFMS will be using Jaeger, which will leverage the Istio sidecar (already part of the microservice Pod) and provide a end-to-end transaction view of a running process.

Monitoring

IFMS 3.0 will be designed on microservice architecture, which will have many moving parts of the application unlike a monolith application. To monitor different aspects of the microservices like container health, application status, load etc, Prometheus will be deployed which consumes different endpoints exposed by the microservice by default and store different metrics parameters like container CPU, memory, etc. These metrics parameter will be further used by Grafana to provide a visual dashboard and also to configure any alarms or alert notification.

5.2. Data Management Requirements

Operational Store (Partitioned RDBMS): This would contain the operational data of the user registration, license key and other configuration related data such as tax records, payments and case information. The data should be horizontally scalable to accommodate multifold times of the initial estimates, partitioned and shared across multiple nodes. The partitioning can be both at row and column levels. Confidentiality would be maintained through partitioning and encryption at a column level for specific columns that are configured to be sensitive. All features such as partitioning, encryption and hashing should be performed at the application level and should not be proprietary features of the RDBMS.

It is essential that the entire system is architected to work in parallel with appropriate data and system partitioning. Data partitioning (or sharing) is integral to ensure that as the data and volume grow, the system can continue to scale without having bottlenecks at the data access level. The architecture must also support mechanisms for read only transactions against a replica instance while insert/update transactions on primary instance. All operational database reads must be centralized via APIs with access and audit.

File Store: The AP could create a separate file storage to store the documents in a more convenient way to ensure that files are easy to search. Existing Eco System eVault will be leveraged.

Analytics & MIS Store: Over a time period, the data being collected and stored by the IFMS Solution will accumulate to provide a wealth of information and insights. These contribute to useful analysis for compliance as well as to demonstrate interesting patterns and exceptions. This would provide necessary inputs and enable the IFMS policy makers to derive useful insights and information. These insights and information can further be utilized to enhance business rules/processes within the IFMS Solution.

5.3. API management

All services of the IFMS Solution will be available through an API layer. Therefore, an API gateway solution is required for it.

The following are some of the expected service mesh/ gateway features for the API management.

1. API layer would not be exposed to an untrusted connection
2. The IFMS Solution must expose APIs publicly on the internet as it is required for the Mobile app. Therefore, all the protection required for public APIs. Any IFMS API which needs to be exposed to external world will be through the state ESB (Raj SewaDwaar)
3. All data transfer must happen through APIs as file transfer mechanism is not encouraged
4. API signature authentication will be through the license key + time stamp + API version and other metadata (including source authentication)
5. All APIs would be stateless in nature, thus easy to load balance, even if hit through the portal is very high and requires high end processing
6. The API Platform should be allowed to manage all enterprise initiatives from a single solution
7. The API Platform should provide clustering and ensure reliability, scalability and single point of administration.
8. The API Platform should provide for enterprise grade encryption
9. The API platform should provide secure access to all APIs and provide all the forms of authentication, authorization, access control and certificate/credential support
10. The API platform should provide a comprehensive threat protection for all API traffic
11. Data sent through the API framework should be digitally signed
12. An API design document with the specification would be shared with the stakeholders for them to start developing the interfaces. The APIs would be RESTful services with XML/JSON payload and would have the minimum information in the design document, purpose of the API, input and output parameter, error code, owners, etc.
13. Version control and API retirement should be taken into consideration
14. Features such as Audit trail, API usage, and API metering should be provided

5.4. Web Portal

IFMS envisages to revamp the existing architecture to make it more scalable and easier to use with the main objective of enhancing the end user experience.

1. The portal should not allow concurrent sessions for same user. The system should automatically log out a customer in case of session breakdowns (e.g., communication failure, high inactivity period - these should be parameterized)
2. The portal should implement security features such as captcha, password complexity, automatic blocking (temporary/permanent) of user logins after given number of unsuccessful attempts.
3. It should by its own or through an integrated Identity Management solution and should be capable of managing security rights and privileges by individual, group and role.
4. Portal should support HTTPS protocol on Secure Socket Layer (SSL)
5. The portal should support the leading browsers such IE, Edge, Chrome, Firefox at the least.
6. The portal should provide search engine with advanced full text search capabilities. The search engine should be able to search for requests within the portal.
7. Portal should be compatible to popular mobile device OS
8. Portal should be interoperable with industry standard databases
9. In addition, the portal should provide the following capabilities
10. Should have multilingual capabilities with regional, localization and Unicode support. (English/Hindi)
11. Should be able to integrate with common office application
12. Should support virtualization
13. Should integrate with standard email services and instant messaging services
14. Should integrate with any other portal products through open standards such as HTML, XML, RSS, web services.
15. Should support encryption and compression features
16. Should support multiple roles with associated access controls.
17. Users should be able to upload documents in specified formats
18. Users should be able to upload multiple files at the same time
19. User dashboard should be mobile responsive
20. IFMS Solution application shall be built on 3-tier architecture with separate presentation, business and database layer

5.5.Microservices

It is envisaged that the IFMS Solution application be revamped, enabling a microservices based architecture / software development approach, which has the distinctive advantage of scalability, agility, higher development productivity, easier debugging and code maintenance.

1. Solution application based on microservices based architecture / software development approach will be developed by the team.
2. All the current features and functionalities of the application shall be maintained until specific request is made by IFMS to remove any such features or functionality
3. The development team shall undertake a re-design effort to change the look and feel of the application or add/ remove features and functionalities as and when desired by IFMS
4. Adoption of Continuous integration Continuous Deployment (CI/CD) practices
5. IFMS Solution application revamp shall also entail container-based deployment of these microservices

5.6.Mobile

Apart for web portal the IFMS application should also be available in the mobile version. Some of the key requirements related to Mobile application, but not limited to, are mentioned below:

1. The Mobile Application should provide an intuitive and user-friendly GUI that enables users to navigate and apply actions with ease. The GUI should be responsive with very little or no delays or time lag at launch or whilst navigating through screens.
2. It should enable ease of configuration and changes to existing GUIs and support the introduction of new screens.
3. It should provide on screen tips and online help to aid users while interacting with it.
4. Should make use of data available in the existing database and reduce duplicate data entry
5. Incorporate analytics into mobile app, to track and identify users experience and actions.
6. Apps should be easily customizable and easy to Administer data in the IFMS database
7. Network level security, traffic should be encrypted using secured connectivity
8. Should structure overall content with proper tagging to make them screen reader friendly.
9. Application should ensure Compatibility with all platforms such as windows, Android, & Mac iOS etc.
10. Solution should develop resolution independent design structure i.e. Mobile Application should adjust itself automatically as per the screen resolution of the Mobile
11. Mobile Apps should work flawlessly across different platforms
12. There should be minimum use flash contents so that home page should be loaded quickly
13. It should not occupy excess client's Mobile RAM.
14. Should provide Role Based Access control
15. Should be able to capture and track all events at device and console.
16. Should come with mobile threat prevention and recovery system
17. Should support authentication using digital signatures

5.7. Integration - Internal/external System

APIs are required to be built for connectivity with 3rd party and internal applications. It is mandatory that the requirements of 'Integration' and 'Scalability' must be considered while developing / customizing the application so that any change is easily addressed as and when these systems are implemented at the state and national level.

SMS Gateway

SMS services are envisaged to be made available as part of the solution design. SMS services can be used for OTP authentication and as well as informing the users about their registration status, payment alert, return update etc. It is a mandatory requirement that all the SMS based services (alerts and notifications) should be available as part of the solution. For outbound message SMS gateway can be developed.

Following are some of the key requirements for the SMS services through the solution:

1. The gateway must be as per prevailing TRAI/DoIT norms. GoR ecosystem has eSanchar, which will be leveraged by IFMS as SMS gateway.
2. Should contain required details / information and targeted to the applicant or designated officers of IFMS departments and other stakeholders and
3. Support automated alerts that allows to set up triggers that will automatically send out reminders

4. Resend the SMS in case of failure of the message
5. Should be instantaneous with almost no waiting time.
6. Must have common features like non-acceptance of landline nos., unacceptable mobile nos. etc.
7. Should Support for Long text messages
8. The message shall be sent though command line interface/API, Web Interface provided by the Service Provider.
9. The vendor shall maintain DND controls.
10. Should provide standard reports like success/failure report on current as well as historical/cumulative basis.

Email service

11. Email services are envisaged to be made available as part of the solution design to send alerts / intimations / automated messages to register email ids, based on preferences set up / opted by individual users. An authenticated SMTP mail service (also known as a SMTP relay or smart host) is envisaged to be integrated with the solution for sending mail from the solution and delivered to intended inbox. Support antispam features. GoR ecosystem has eSanchar, which will be leveraged by IFMS as email gateway.

6. API Framework

IFMS Solution will be an API based solution Team will also build APIs. IFMS will be the overall regulator and overseer of the API based system.

Following are some of the key principles for API framework

1. API layer would not be exposed to untrusted connection
2. All external users will connect to solution/portal through SSL Layer of authentication along with user id, single sign authentication / OTP etc.
3. All APIs level access either to department systems or to app providers servers should be through HTTPS
4. All data transfer from / to the system to happen through APIs
5. All the APIs would be stateless in nature, thus easy to load balance, even if hit through portal is very high and this requires high end processing.
6. Team would deploy a developer sandbox to test the APIs with dummy data.
7. An API design document with the specification would share details to start developing the interfaces. The APIs would be RESTful services with XML payload and would have the following minimum information in the design document.
   a) Purpose of API
   b) Author & Owner of API (controlling entity)
   c) Input parameters
   d) Output
   e) Error codes

Since IFMS Solution will be an API based solution where external agencies / eco system partners will also build & manage APIs as well as will set up secured to access the system. Stakeholder can access System through these agencies also apart from accessing the services through portal. Hence, it is important that the Team sets up, manage and monitor the API services for proper operation of system. This entire framework needs to be set up and operationalize as part of the application development. The broad scope of the Team relating to this function is detailed below:

6.1. Enrollment and operations

IFMS will be the overall regulator and overseer of the API based system. As part of this project, Team will set up the requisite process as well as system to build, operate & manage and sustain APIs in a secured and controlled environment. These system partners will establish secure connectivity compliant with IFMS's standards and specifications. The external partners will offer their IFMS-compliant network connectivity as a service and transmit authentication requests to the system.
Only application partner contracted with IFMS as external system partners shall send authentication requests to the solution; no other entity can directly communicate.
The system partners will use IFMS authentication to enable its services

6.2. Standardizing API and specification

Based on the solution developed API needs to be standardized and specification has to be defined so that it can be exposed in the system partners ecosystem and system partners can create their own services and expose them to the outer world for stakeholder use. Team should have the ability to define, document, edit and publish API specifications.

6.3.Environment Management

Team should provide a sandbox environment where system partners can publish a mock version of APIs. The admin portal shall be used to create partners Dev IDs that can be accessed by the developers of potential system partners for development, test and integration. User can perform testing in a sandbox environment which is distinct from production. Sandbox shall provide the same catalogue as the production framework; however, these APIs are stubbed/ mocked only. All the APIs shall be hosted in sand box environment to ensure at-least couple of system partners integrate/test before the API is moved to production.

6.4. User Authentication

Team to provide authentication services for allowing users to access the above-mentioned environments and to do the following operations

1. To authenticate user into the Sandbox.
2. To configure authorization policy for new APIs as they are introduced to the framework.
3. Allow user access the available APIs and associated properties in accordance with his/her entitlements.
4. Allow Client app exposed to the API, resources data in accordance with the configuration for that app.
5. Blacklist/Block Access

6.5. Publishing and Management of API

There should be a mechanism that allows authorized users to publish new APIs as they are created to sandbox, test and production environments as required. Once the API are developed and deployed in the sandbox environment Team to do a proper functional, security and performance test and certify the API before been published for production usages. An API catalogue should be maintained by the system.

6.6.Version Control

Team should provide a controlled mechanism for API versioning control for any change. The version & release management process must cover this aspect to ensure any change is made or rolled-out in a controlled & informed manner.

6.7. API Retirement

Team should provide a mechanism to retire/archive APIs. The solution should provide full support of managing retire and archiving APIs as part of the life cycle and associated version control.

6.8. API Governance & SLA enforcement

Team should provide a mechanism to define and enforce SLAs/quotas for consuming entities of the API framework. The solution should provide “Plan” such as to control how much traffic can be sent by each user through the interface. A Plan to make available a collection of resources from one or more APIs, or a plan defines a rate limiting policy that specifies how many requests an application is allowed to make during a specified time interval, and what action should be taken when the threshold is exceeded. The solution should support both a hard limit which throttle the traffic and a soft limit which notify the administrator about the policy violation. For soft limit the solution should be further customized to buffer the extra messages to shape the traffic. Application developers gain access to APIs by registering their applications to access plans by using the developer portal.

Solution should also provide real time SLA monitoring capability. Solution should enable configuration of system SLAs. The APIs load shall be continuously and pro-actively monitored for suitable & prompt actions in case of excessive loads, failures or performance bottlenecks. Administrator should perform appropriate actions as SLA thresholds are approached/ reached - including notifications, throttling, rejection of requests, raising alarms.

API framework and IFMS Solution must work together to synchronize API and back-end service policies. Additionally, API lifecycle stages should synchronize with system service implementation stages. An API Governance experience may provide a straightforward set of lifecycle stages (e.g., created, published, deprecated, retired, blocked) that may be customized by the development team.

6.9. API Updates, Notification and tech support

Team should provide a mechanism to provide consuming entities with appropriate notifications with respect to APIs. The solution should ensure that application developers who wish to access and use the APIs are self-sufficient. Documentation about an API, such as URL used to call the API and the security mechanism used by the API to authenticate application user, is automatically generated when defining the API and exposed through the developer portal. Additional supporting documentation that can further help application developer to use the API, such as samples and/or tutorials and other supporting documentation shall be available through the developer portal.

6.10. API Security Governance

Solution should provide appropriate & adequate security mechanisms governing access to API framework- i.e., Authentication strength. The system should inspect the headers for APIs genuineness before acceptance. It should apply all security checks e.g. DDoS Attacks, XML Denial of Service (xDoS), Slow down or disable an XML based System, Message Snooping, XML Document Size Attacks, XML Document Width Attacks, XML Document Depth Attacks, Jumbo Payloads, Recursive Elements, Public Key DoS, XML Flood, Resource Hijack etc. to ensure rightful and secured access to API consumers.

Assume that solution shall require security architecture approval; solution must adhere to IFMS security protocols/standards.

6.11. Audit Trail

Solution should allow the audit tracking of dev/client apps consuming APIs and portal API. Audit log should have capability to track (not an exhaustive list)-

1. User ID/Credential
2. Transaction data
3. IP addresses of source and destination
4. Date & time stamp
5. Session duration
6. Session ID/key (Session key must be dynamically generated for every transaction session key must not be reused)
7. License Key

6.12. API Usage Activity

Solution should provide with reporting data on the usage/transactions performed by users/consuming entities on exposed API. The solution should be capable of generating reports/Online view that shows API usage information such as most/least accessed APIs, highest/lowest/average response time, data volumes transferred, transfer rates, the maximum and minimum response time over a period of time, rejected API requests. Custom queries can be created to further refine the API usage information such as identifying API calls that falls within a window of response time.

7. Security

Security is one of the utmost important aspects envisaged in the entire solution design of IFMS Solution. All key dimensions both from proactive and reactive security measures including authentication, authorization, access control, logging and monitoring should be an integral part of the system architecture. Enterprise level solutions will be part of Infrastructure RFP, the Team needs to use the solutions provided by IFMS to ensure end to end security.

The key security requirements are mentioned below:

1. Should support encryption for all messaging components including local store of data. Local Store encryption shall be customizable for various levels of encryption required. Encryption shall not mean password protection.
2. Should provide inbuilt support for digital signature and PKI based on industry standards such as X509 v3.
3. Should provide support for simple, flexible administration via a web browser and thick client
4. Should support TLS/SSL encryption with 256/512/1024 bit key.
5. Should be able to perform Anti-Relay enforcement on incoming connection, allow only the customers domains.
6. Should provide integrated PKI as a foundation for numerous security features, including digital signatures and encryption; granular access control - down to the individual field level; execution control lists; local data encryption; and trust relationships in multi-organization and Extranet applications.
7. Should allow multi-level passwords to ensure that no one administrator may have full control of the important user credentials.
8. Should provide support for field level security for all messaging components.
9. Should provide Reader control at the document/record level to ascertain only selective users may ever be able to search and access such secure and confidential content.
10. Critical aspect of exposing the API is around the security policy to be applied. The enterprise security policy could be beyond API level security & tie into various other layers such as App, transport etc. In addition, the security aspect also includes making the API gateway tamper proof with use of PKI (Public key infrastructure) using HSM card to have hardware key and lock mechanism security.
11. Public key cryptography is at the heart of XML Web services security. XKMS (XML Key Management Specification) is designed to simplify the integration of PKI and digital certificates to enable authentication, digital signature, and encryption services, such as certificate processing and revocation status-checking, without the complications historically associated with proprietary PKI software toolkits.

Other required features are mentioned as below:

7.1. Authentication and Authorization

IFMS solution should have digital signature implemented in the system from start of the project. Integration of e-Sign which is currently being envisaged by Government of India also needs to be integrated with the system as on when the same become operational.

1. The system should comply with all requirements of security, reliability and non-repudiation as per the Government of India guidelines. For the Portal there should be a provision of logging

into the system through Internet as well as Intranet. There should be provision for authentication using digital certificates as per the government of India guidelines. Center / State User will however reserve the right to procure digital certificate for the department and end users whenever required.

2. Once the users enter their login credentials, the user credentials from the user authentication server database must be verified and then only the access should be granted inside the system.
3. The logged in users with the adequate privileges, as granted through the 'Admin' module, should be able to access the modules in the system.
4. The system should provide the single sign-on facility i.e. once any Users credentials are verified, he / she should be able to navigate through all the modules and functionalities of the integrated application, to which that User is authorized to access, based on Role Based Access control (RBAC).
5. The security solution must monitor all traffic to all resources of the system and all access attempts to the system or directly to any resource managed/access by the system, should be intercepted by the security solution, and analyzed for authentication and authorization requirements defined for the resource.
6. Any access to end users to database should only be via application/API authorization.
7. Solution should allow a user to access various functions, forms, screens, sub modules, information, etc. as per the authorization and user role permitted by the system administrator as per available guidelines and policies.
8. Public user can browse the portal with rights to view public content available on the website, remaining all types of users shall enter the solution using appropriate secured authorizations.
9. The proposed solution shall support flexible definition and modification of authentication, authorization, encryption, and data integrity assertions and requirements for each security policy
10. The solution should support multiple authentication methods such as Username password, two factor authentication, and digital certificate.
11. The solution should have the functionality to provide authentication based on the role.
12. The solution should not store authentication credentials on client computers after a session terminates.
13. Should support authentication – SMTP AUTH, POP before SMTP, File system, Database, LDAP etc.
14. Authentication should be done for all valid users. A valid User for this application should be the one who has been set-up in the application such that he/she can access the application and perform tasks as per assigned roles and responsibilities as well we access rights within the system
15. The system should have a configured directory of all authorized Users.
16. User Management should be a management and authentication feature within the application that should provide administrators with the ability to identify and control the state of users that should have right to log into the "IFMS Solution" and take or provide services through it.
17. There should be a form of User authentication functionality that should allow various users to access the "IFMS Solution" and work as per their defined Roles and Responsibilities. Access Management should be used for restricting access to rights-protected content / sections / modules / screens / Fields, etc. to authorized users only. Rights to all active users should be granted based on their hierarchy and level in the organization, designation, assigned roles and responsibilities, location etc. among other parameters. It is also proposed that the new privileges can be created through the access Management UI interface as well as existing rights be managed through the same. The access to the section of the application should be strictly based on "Role Based Access Control" (RBAC) for the Administrator(s) only as defined in the ACCESS Policy. The details of any change in the module should be captured in the

Audit Trail of the application. Also, there should be facility to assign/modify/deactivate/delete rights globally for the desired Groups within the system.

18. OAuth functionality should be enabled for Open API to enable access delegation through secure servers.

It is critical to identify the consumer of the API to validate the caller by providing a client id & secret which needs to be embedded in each request. In addition, the end user of the app may also need to be authenticated or authorized to access the resources through a username/password. This can be typically done through Basic Auth or OAuth as described above. The directory against where it authenticates could be standard LDAP or non-standard user registries.

7.2. User and Identity Management

1. The solution should be capable of uniquely identifying all users of the system and their activities.
2. The solution should have the capability of providing user access rights to system and data which will be in line with the defined functional requirements.
3. The identified solution should support both on premise and cloud implementation, or a hybrid of the two.
4. The solution should be a directory-based solution supporting LDAP.
5. The user account management component of the solution should address requesting, establishing, issuing, suspending, modifying, and closing user accounts and related privileges, with a proper approval process.
6. The system should be able to perform regular audits and management reviews of all accounts and related privileges.
7. Should provide Single Sign-On facility.
8. Should support mobility.
9. Should support privilege user management (Super user, Admin accounts).

7.3. Application Security

1. IFMS Solution must comply with the Application Security Plan and security guidelines of Government of India / IFMS as applicable
2. Validation checks should be incorporated into the application to detect any corruption of information through processing errors or deliberate acts.
3. Data output from an application should be validated to ensure that the processing of stored information is correct and appropriate to the circumstances
4. Should implement secure error handling practices in the application
5. IFMS Solution should have Role based access, encryption of user credentials. Application-level security should be provided through leading practices and standards including the following
   f) Prevent SQL Injection Vulnerabilities for attack on database
   g) Prevent XSS Vulnerabilities to extract username password (Escape All Untrusted Data in HTML Contexts and Use Positive Input Validation)

h) Secure Authentication and Session Management control functionality shall be provided through a Centralize Authentication and Session Management Controls and Protect Session IDs from XSS
i) Prevent Security Misconfiguration Vulnerabilities (Automated scanners shall be used for detecting missing patches, misconfigurations, use of default accounts, unnecessary services, etc. maintain Audits for updates
j) Prevent Insecure Cryptographic Storage Vulnerabilities (by encrypt off-site backups, ensure proper key storage and management to protect keys and passwords, using a strong algorithm)
k) Prevent Failure to Restrict URL Access Vulnerabilities (By providing authentication and authorization for each sensitive page, use role-based authentication and authorization and make authentication and authorization policies configurable
l) Prevent Insufficient Transport Layer Protection Vulnerabilities (enable SSL for all sensitive pages, set the secure flag on all sensitive cookies and secure backend connections
m) Prevent Id Redirects and Forwards Vulnerabilities
n) For effective prevention of SQL injection vulnerabilities, IFMS should have monitoring feature of database activity on the network and should have reporting mechanism to restrict or allow the traffic based on defined policies.

7.4. Data Encryption & Object Signing

1. All the interfaces between various applications and user are encrypted using appropriate protocols (such as HTTPS, IPSec, SSL etc.), algorithm and key pairs.
2. System should support 128/256/512-bit encryption for transmission of the data over the Internet.
3. Object signing and encryption of attachments (documents) should be compliant to published authority standards (like DeitY).
4. Proposed solution must be secured to both internal and external parties (such as through encryption)
5. The Network / Transport level should include Network Link Encryption (IPSEC) and encrypted HTTP session using TLS/SSL (HTTPS)
6. Business data should be encrypted in the database and DBA should not be able to read or modify it.
7. Audit controls, electronic signatures, data encryption and other methods should be used to assure the authenticity of transaction and other relevant data
8. Following events should be considered as security incidents: unsuccessful log-on, intrusion detection, malfunctioning of encryption facility, etc.
9. Should develop a procedure for archiving the log files and ensure security of the log files
10. Separate environment should be maintained for production, test and development to reduce the risks of unauthorized access or changes
11. System should have the functionality to record all the administrator, user level activities including the failed attempts
12. Should protect logging facilities and log information against tampering and unauthorized access
13. Information security baseline document should be developed for all the infrastructure components such as database, operating system, router, switch etc. based on CERT-In technical guidelines and best practices.
14. Provisions should be made for secure content management.
15. Solution should ensure logs including at least the following:

a) Authentication and Authorization events – logging in, logging out, failed logins. These should include date/time, success/failure, and resources being authorized, the user requesting the authorization and IP address or location of the authentication attempt
b) Logs for deletion of any data
c) Logs of all administrator activity
d) Logs of modification to data characteristics: permissions, location, field type

.

7.5. Data Integrity

Data in transit (from external systems or between internal systems) or data at rest must be protected from tampering. The risk is from both external users and internal users (such as Database Administrators) who are close to the data at all times.
To handle the risks of data being tampered by the external users and during transit, API design must ensure checksum features and digital signatures to validate the data is secured. The API documents explain these features in detail and all the sensitive data must adhere to these principles. The system shall ensure to validate integrity using the checksum and digital signature validations before processing the data.
To handle the risks of data being tampered by the internal users such as Database Administrators who have access to data, IFMS Solution shall be designed with the below principles:

1. All the data access must be enabled only through internal API / modules.
2. Each subject area can be packages into a persistence module that exposes domain specific methods to read, insert, update or delete the records
3. Persistence module shall abstract the underlying data base technologies, physical data models

7.6. Data Confidentiality

To ensure data is secured to access only by required teams and applications, the following principles are to be adhered:

1. All the databases must be accessed by individual user accounts and user accounts cannot be shared by multiple persons or as a team-based accounts.
2. All the databases/systems must be integrated with the Identity Access Management system for centralized control. This will also enable disabling of user accounts when a person leaves the organization.
3. For reporting purposes, the data MUST be anonymized (e.g. user level id and details are to be masked- mobile number can be stored as a hash value etc.) before publishing to the BI reporting system.
   a) Any ID must be stored by a UUID value that cannot be easily guessed.
   b) The mapping of such critical id's can be stored as master information in main IFMS databases.
   c) Any other sensitive information must be hashed and stored in BI system

4. Sensitive data stored in the main RDBMS tables must be encrypted so that Database administrators do not have direct access to this information for misuse

a) Encryption must be done on similar lines of data integrity. However, to ensure no significant performance loss, the Hardware security modules must be used to encrypt and decrypt the data while persisting and reading the data.
b) All applications reading the data must use the common persistence modules designed for each subject area to abstract the implementation complexity and expose the solution as a re-usable component.
c) The keys used to encrypt the data shall be critical information that must be protected at all times.

7.7. Message Protection and Integrity

The Request message JSON / XMLs would be protected using HMAC. The HMAC would be used to simultaneously verify both the data integrity and the authentication of a message.

7.8. Digitally signed requests

In some cases, the Request JSON / XMLs would be digitally signed that would help in the non-repudiation of the requests. The system would also send digitally signed XMLs as response for specific cases. The XML message received as input for some of the message can be validated with configuration.

8. Monitoring

The whole system is expected to meet expected performance levels with near 100% uptime through various fault tolerance/BCP/DR practices envisaged in the overall solution. As part of sustenance, the Team shall integrate with all required tools provided by IFMS to pro-actively monitor & manage the whole setup and deliver the committed SLAs. The Team will ensure that the reports for monitoring the SLAs like uptimes, system performance; Key performance indicators etc. are generated automatically from the system and made available to IFMS. Some of the key requirements for these services are mentioned below:

1. Automatic monitoring of overall service availability as well as each individual component of the system including all Infrastructure and Software components, services in a co-related fashion
2. Deep analytical capabilities e.g., Application layer traffic analysis for networks
3. Should provide tools and metrics to support testing, solution performance monitoring, fault isolation, verification and validation of the end-to-end solution.
4. Should have the ability to monitor in real-time all the activities and transactions of all the solution components.
5. Should have the ability to show the status of all components, process and all components through establishment of Child-parent relationship
6. Should have the ability to show all the services running across the solution and the recent reference bindings with an actual view of the service flows showing service-to-service relationship as well as drill down to service specific information.
7. Should have the ability to monitor and show all the status of the different infrastructure layers supporting the solution platform.
8. Should have the ability to monitor & report the performance of the various subsystems.
9. Should have the ability to show recent faults and errors and be able to display recent error messages and exceptions handled.
10. Should be able to auto-ticket for pre-defined major alerts
11. Should support meeting specific SLA’s by issuing customized and configurable alerts.
12. Should provide visualization of the overall system.
13. Should be able to produce online reports/dashboard showing various availability & performance metrics.
14. Should provide reports to authorized users for end-to-end performance monitoring and control.

9. Software Development Lifecycle

9.1. Continuous Integration & Continues Delivery

The system development should be highly modular and parallel development should be carried out for faster execution using industry's best Software Development Lifecycle practices. All application modules within the same technology platform should follow a standardized build and deployment process.

A dedicated 'development / customization' environment should be proposed and setup. The team must provision separate development and testing environment for application development and testing. Any change, modifications in any module must follow industry standard processes like change management, version control and release management in large and complex application development environment.

Application source code could be maintained in source control and could be broken up into a number of projects. Source control projects are created to abstract related set of modules or feature that can be independently included in another application.
It is a mandatory to create, update and maintain all relevant documentation throughout the contract duration. Also, it should be ensured that a bug tracking toll is maintained for proper tracking of all bugs fixes as per various tests conducted on the application

9.2. Container Architecture

For development and deployment, the container-based architecture is proposed for seamless application development & deployment.

Team should use a Container Architecture tool for entire development life cycle e.g. developing, shipping, and running applications. With container Architecture, the developer teams can separate the applications from the infrastructure and treat the infrastructure like a managed application. It can also help to ship code faster, test faster, deploy faster, and shorten the cycle between writing code and running code. It allows the developers to develop on local containers that contain the applications and services. It can then integrate into a continuous integration and deployment workflow.

Following objectives can be achieved by a Container Architecture:

1. Faster delivery of the applications
2. Easy Deployment and scaling
3. Achieving higher density and running more workloads

9.3. Quality Assurance

A thorough quality check is proposed for the system and its modules, as per standard Software Development Life Cycle (SDLC). Team is expected to lay down a robust Quality Assurance

program for testing of the developed application for its functionality, performance and security before putting in production environment. The program must include an overall plan for testing and acceptance of system, in which specific methods and steps should be clearly indicated and approved by IFMS. Team is required to incorporate all suggestions / feedback provided after the elaborate testing of system, within a pre-defined, mutually agreed timeline. TEam must undertake the following:

1. Outline the methodology that will be used for testing the system.
2. Define the various levels or types of testing that will be performed for system.
3. Provide necessary checklist/documentation that will be required for testing the system.
4. Describe any technique that will be used for testing the system.
5. Describe how the testing methodology will conform to the requirements of each of the functionalities and expected outcome.
6. Indicate / demonstrate to IFMS that all applications installed in the system have been tested.

9.4. Automated Testing

Team is expected to perform automated testing with following features:

1. Should support multi-layer test scenarios with a single solution.
2. Should support and execute testing on GUI and UI-Less (standard Web Services, non-SOAP Web Services, such as REST, etc.) Components.
3. Should allow version control of tests and test assets providing ability to compare versions and identify changes.
4. Should allow centralized storage and management of tests and test assets including external resources used by tests.
5. Should have an IDE environment for QA engineers which should be configurable.
6. Should provide local system monitoring to test and validate performance issues including memory leakage, CPU overload and network overload to determine if specific business scenarios exceed desired performance thresholds.
7. Should provide Auto-documentation while creating of automated tests.
8. Should generate reports that can diagnose defects and can be exported to PDF, DOC and HTML formats.
9. Report with summary data, pie charts and statistics for both the current and previous runs needs to be provided.
10. Should enable thorough validation of applications through a full complement of checkpoints such as GUI object, database, XML, XPath, etc.
11. Should provide Unicode support for multilingual application testing.
12. Should be able to record the test Execution into a video file for viewing later.
13. Should provide facility to parameterize tests to generate/assign test case output values automatically during runtime.

9.5. Performance and Load Testing

Team is expected to implement performance and load testing with following features:

1. Testing workload profiles and test scenarios based on the various functional requirements should be defined. Application as well as system resource utilization parameters that need to be monitored and captured for each run also needs to be defined.

2. Should support application testing and API testing including HTTP(s), web services, mobile applications and different web 2.0 frameworks such as Ajax/Flex/HTML5.
3. Team should perform the load testing of application for multiple workload profiles, multiple scenarios, and user loads to handle the envisaged users of the system.
4. Different activities before load testing i.e. identification of workload profiles, scenarios, information capturing report formats, creation of testing scripts, infrastructure detailing and workload profile should be prepared before the start of actual load testing exercise.
5. Solution parameters needs to be tuned based on the analysis of the load testing reports. The tuning process could be iterative until the issues are closed. Multiple load runs need to be executed for users to simulate different scenarios, such as peak load (year end, quarter end, etc.), load generation within the LAN, Load generation across WAN or mobile network simulator while introducing configurable latency/jitter/packet loss etc.
6. Should eliminate manual data manipulation and enable ease of creating data-driven tests.
7. Should provide capability to emulate true concurrent transactions.
8. Should identify root cause of performance issues at application or code level. Include code performance analysis to quickly pinpoint component-level bottlenecks: Slowest classes and methods, most frequently called methods, most costly (aggregate time spent for each method), response time variance etc.
9. Should allow selection of different network bandwidth such as analog modems, ISDN, DSL, or custom bandwidth.
10. Should be able to monitor various system components e.g., Server (OS, Web, Application & Database) Monitoring, Network (between Client & Server) Delay Monitoring, Network Devices (Firewall, Switch & Router) Monitoring during the load test without having to install any data capturing agents on the monitored servers/components
11. Should correlate response times and system performance metrics to provide quick insights into root cause of performance issues.
12. Reports on following parameters (but not limited to) such as transaction response time, transaction per second (Passed), user interface rendering time, transaction per second (Failed), web transaction breakdown graphs, hits per second, throughput, HTTP responses per Second, pages downloaded per second, system infrastructure performance metrics etc.
13. Should provide End-to-End system performance analysis based on defined SLAs. Should monitor resource utilization including memory leakage, CPU overload and network overload. Should have the ability to split end-to-end response time for Network & Server(s) and provide drill-down capability to identify and isolate bottlenecks.

10. Data backup and recovery

10.1. Data Backup Standard

- Critical data must be defined by GoR and must be backed up.
- Backup data must be stored at a backup location that is physically different from its original creation and usage location (i.e., The Disaster Recovery Site).
- Data restores must at least be tested quarterly.
- The GoR must document procedures for backing up critical data and the testing of the procedures. These procedures must include, as a minimum, for each type of data and system:
    - A definition of the specific data to be backed up.
    - The type(s) of backup to be used (e.g., full back up, incremental backup, etc.).
    - The frequency and time of data backup.
    - The number of generations of backed up data that are to be maintained (both onsite and offsite).
    - Responsibility for data backup.
    - The storage site(s) for the backups.
    - The storage media to be used.
    - Any requirements concerning the data backup archives.
    - Transport modes; and
    - Recovery procedure of backed up data.

10.2. Data backup selection

All data and software essential to the continued operation of the Portal, and all data that must be maintained for legislative purposes, must be backed up. All supporting material required to process the information must be backed up as well. This includes programs; control files, install files, and operating system software. The GoR will determine what information must be backed up, in what form, and how often.

Backup types

- Full backups should be run weekly as these datasets will be stored for a longer period. This will also help ensure that data can be recovered with the minimal set of media used at that time. Once a month, a full backup should be stored off-site. This statement will be subject to the DR Business Impact and Risk Analysis requirements review with input from System Administrators and GoR operators.
- Differential/Incremental backups must be used for daily backups. This ensures that the backup time window is kept to a minimum during the week while allowing maximum data protection.

- If a system requires a high degree of skill to recover from backup, consideration must be given to making full images of such servers as a backup. This will ensure that the system can be retrieved with minimal knowledge of the system configuration.

10.3. Backup schedule

- Backup schedules must not interfere with day-to-day operations. This includes any end of day operations on the Portal.
- A longer backup window might be required, depending on the type of backups
- When the data on the Portal changes frequently, backups need to be taken more regularly to ensure that data can be recovered in the event of a system failure.
- Immediate full data backups are recommended when data is changed to a large extent or the entire database needs to be made available at specific points in time.
- Regular, as well as event-dependent intervals need to be defined.
- The GoR should determine the number of previous versions of operating systems and portals that must be retained at the Backup and Disaster Recovery location.
- Annual, monthly and weekly backups must be retained at the Backup and Disaster Recovery location. Weekly, Monthly and Annual backup media may be re-used to take new backups.

10.4. Data backup procedures

The GoR has the discretion to choose between automated and manual backup procedures based on their requirements and constraints. Both procedures are in line with best practice.

The table below outlines the two procedures with their advantages and disadvantages:

Type	Advantages	Disadvantages
Manual backups Manual triggering of the backup procedures or manual process of transporting backup media.	• The operator can individually select the interval of data backup based on the work schedule.	• The effectiveness of the data backup is dependent on the discipline and motivation of the operator. • Higher risk of backup media loss of integrity. • Requires more resources to be effective.
Automatic backups Triggered by a program at specific intervals. The entire process is electronic with no manual operations required.	• The backup schedule is not dependent on the discipline and reliability of an operator.	• There is a cost associated with automation. • The schedule needs to be monitored and revised to include any non-standard

	• Lower risk of backup loss or media integrity. • Fewer resources required.	updates and/or changes to the work schedule.

The GoR has the discretion to choose between centralised and decentralised backup procedures based on their requirements and constraints. Both procedures are in line with best practice. The table below outlines the two procedures with their advantages and disadvantages:

Type	Advantages	Disadvantages
Centralised backups The storage location and the data backup's performance are, where possible, carried out on a central ICT Backup system.	• Allows for more economical usage of data media.	• There is added exposure to confidential data. • Confidential and non-confidential information may be combined, requiring more stringent security controls for handling the backups.
Decentralised backups Performed by ICT end-users or system administrators may or may not be transferred to a central ICT system.	• ICT users can control the information flow especially in the case of confidential data.	• The consistency of data backup depends on the reliability and skill level of the end-user. • High risk of data exposure or loss.

10.5. Storage medium

When choosing the data media format for backups, it is important to consider the following:

- Time constraints around identifying the data and making the data available.
- Storage capacity.
- Rate of increasing data volume.
- Cost of data backup procedures and tools vs. cost if restored without backup.
- Importance of data.
- Life and reliability of data media.
- Retention schedules; and
- Confidentiality and integrity

10.6. Data backup owner

The GoR should ensure that sufficient ICT capacity is available to maintain the Backup and Disaster Recovery procedures, to ensure a segregation of duties and responsibilities and to mitigate the risk of systems and data losses.
The Portal administrator has the discretion to assign at least two staff members (One primary, one secondary) to ensure each backup schedule is maintained.

10.7. Offsite storage site

- Data backups must be stored in two locations:
    - One on-site, or in close proximity, with current data in a machine-readable format if operating data is lost, damaged or corrupted; and
    - An off-site location to additionally protect against loss to the primary site and on-site data.
- Off-site backups must be a minimum of 5-6 kilometres from the on-site storage area to prevent a single destructive event from destroying all copies of the data.
- The minimum requirements are to store the monthly and/or yearly backup sets off-site.
- The site used for storing data media off-site must be physically secure and safe.
- Should an off-site media set be required to perform a restore, the data media must be returned to the offsite facility for the remainder of the applicable retention period.
- All data media used to store information must be disposed of to ensure the data is not recoverable.

10.8. Transport modes

When choosing the transport mode for the data (logical or physical), it is important to consider the following:

- Time constraints
- Capacity requirements
- Security and encryption.

10.9. Retention considerations

Data should be retained in line with current legislative requirements.
The minimum retention of backups are as follows:

- A full system backup will be performed weekly, where possible or not longer than two (2) weeks. Weekly backups must be saved for a full month.
- The last full backup of the month will be saved as a monthly backup. The backup system will recycle the other weekly backup media.

- Monthly backups must be saved for one year, at which time the media will be reused or disposed of.
- Yearly backups must be retained for five (5) consecutive financial years and will only be run once a year at a predetermined date and time.
- Differential or Incremental backups will be performed daily. The GoR should determine the retention of Daily backups. Daily backup media will be reused once this period ends.

10.10. Recovery of backup data

Backup documentation must be maintained, reviewed, and updated by the Portal administrator periodically to account for new technology, business changes, and Portal migration to alternative platforms. This includes, but is not limited to:

- Identification of critical data and programs; and
- Documentation and support items necessary to perform essential tasks during a recovery process.

Documentation of the restoration process must include:

- Procedures for the recovery
- Provision for key management should the data be encrypted.

Recovery procedures must be tested at least quarterly.
Recovery tests must be documented and submitted to the Portal administrator.

10.11. Data categorization

The data available within different systems are categorised for shareability based upon its sensitivity, granularity, criticality and ownership/data origination. Data falling under the Sensitive category shall ordinarily be provided against a specifically authorised request on a case-by-case basis and backed with a Non-disclosure Agreement between the requestor and acceptor.

Non-sensitive data: Highly summarised or aggregated data created by combining information on individuals and legal entities shall be considered as Non-Sensitive. Data mandated by Statute or other Acts to be publicly displayed or openly hosted on Government portals shall be treated similarly. At its discretion, the GoR could decide to openly publish any data which it feels is required in the interest of transparency or public benefits.

Sensitive data: The following can be categorised as sensitive data:

1. Personal Data includes all data about an individual entity. Sensitive Personal Information (“SPI”) and Personally Identifiable Information (“PII”) data is Highly Sensitive.
2. Commercially sensitive information has financial implications for a taxpayer; if compromised, it can cause economic impact to a taxpaying entity such as invoice details, pricing information, supplier details, etc.
3. Data that comes into existence through enforcement functions of any department; data generated as part of internal analysis using the department's internal tools and

techniques for profiling as part of enforcement functions, risk analysis, investigations and intelligence gathering.

4. Granular data pertaining to an individual's access/activity logs in the system and other forensic data is available in the department's IT systems.
5. Data provided to a department by another Government organisation or any other organisation through an existing arrangement or with whom the department has executed a contract.
6. Third-party granular transactional data in individual import/export documents, returns, payments etc., shall ordinarily be treated as Sensitive data regarding its commercial sensitivity.

10.12. Data Archival

Database stores values from Master data and Transactional data. Day to day transaction increase the size of data in the database which impact performance of the application w.r.t Database. To overcome this Data Archiving Strategy to be in place. Recommendation for Data Archival for Transactional Data.

1. Identification of Tables to have achieve Policy.
2. Date would be the best parameter for transactional data archive record.
3. Automatic Archive schedule/Policy to be in place.
4. Keep only 3-year data on production server and rest to be archived.
5. Accessibility policy of Archive data to be present.
6. Like what are the systems that data would be required
7. Time to retrieve
8. Archive data to be read only.
9. Accessibility methods to be defined with authority and security Key to be in place.
10. Archive Data to be indexed for faster retrieval.
11. Archive information Management details to be documented, which will help to know how many years of data as per Legal / Departmental / Financial Year to be in place.