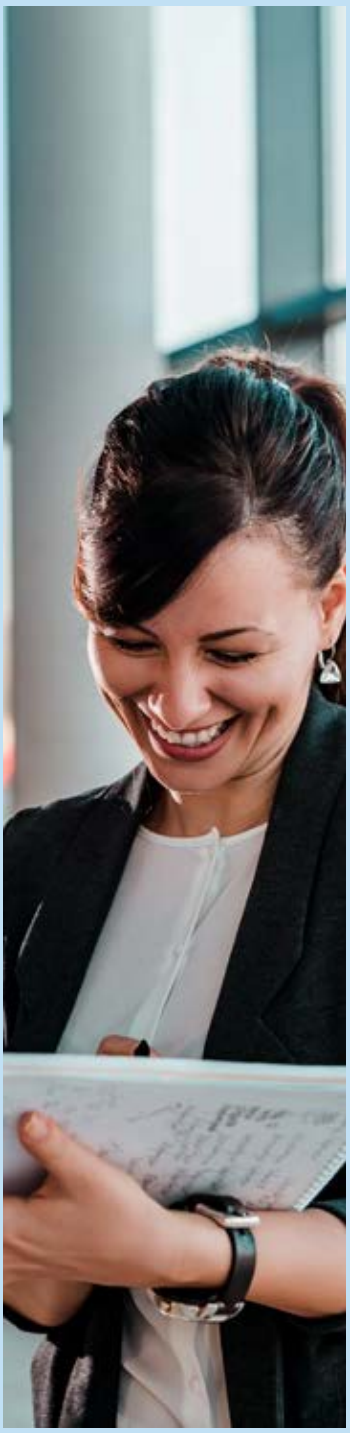
VIEW POINT



CONTENT ARCHIVING WITH INFOARCHIVE

Infosys®
Navigate your next

Executive Summary

Making a decision on archival of legacy applications' data is a difficult but necessary step for many IT organizations today, considering the very high data volume and the resultant costs associated with maintaining such applications. Similarly, live applications with high storage footprint pose cost challenges to IT departments. Regulatory, compliance and legal requirements add further complexity around the data storage strategy. All this, coupled with disparate type of data characteristics present in an organization (e.g. data formats, confidentiality requirements, retention needs, output adaptabilities), as well as increasing digital requirements (e.g. analytics, reporting, integrations) makes selection of an archival platform a non-trivial task.

Due to the sheer volume and resultant costs, unstructured data (i.e. content) archival is one of the hot areas in ECM landscape of many organizations. While there are many archival products available in the market offering a number of options to handle the data complexity, cost and compliance requirements, it can be daunting to zero-in on a cost effective product while ensuring the compliance and performance expectations are met.

OpenText InfoArchive is one of the leading players in the archival landscape, allowing archival options for the data in a cost-effective and compliant manner, while offering the digital features, and flexibility to accommodate any future extensions.

This white paper provides insights into InfoArchive architecture, typical use cases and design considerations for InfoArchive based archiving solutions for content e.g. archive structure, search design, transformation strategy, retention strategy, storage options etc., which in turn, can aid in designing a cost-effective, standards-based, performant, infrastructure-agnostic and compliant archiving platform.

InfoArchive – How It Works

InfoArchive is an enterprise archival platform, based on OAIS (Open Archival Information System) standards. It supports very high volume archiving scenarios (petabytes scale), with structured and/or unstructured data, and enhances the ability to secure and leverage critical application data and content.

It supports multiple languages, enabling the use suited for global audience. Due to its enormous scaling capabilities, it's well suited as an application agnostic archival mechanism, i.e. multiple applications can be archived on a single InfoArchive instance, separated from each other by security controls, and can have different design elements.

Being an OAIS-compliant archival solution, InfoArchive ensures to:

- Gather and accept appropriate information about the data, from information producers
- Apply sufficient controls on the information to ensure long-term preservation
- Determine the access scope of the target user base
- Make the information understandable by the target audience, without any assistance from original information producers
- Follow defined policies and procedures to preserve the data information against reasonable contingencies, and to disseminate the information as authenticated copies of the original data or as traceable to the original
- Make the information available to the target user base

Architecture

InfoArchive has the classic three-tiered architecture : Native XML database, called xDB at the backend for data persistence, InfoArchive server as middleware, for computational duties and InfoArchive web UI, providing user experience.

The newer versions of InfoArchive (4.x onwards, with 16.5 stack being the latest) include a new architecture, shedding the Documentum components present till 3.x. The new InfoArchive application has an AngularJS and Bootstrap based UI, with the server relying on spring Data. The result is an application that is lightweight and more performant.

The REST API layer follows the decoupled architecture paradigm, allowing a custom UI to be built for presentation where need be. It enhances the interoperability capabilities, allowing the functionalities to be used via REST calls, while data encryption capabilities help secure the data at rest.

The modular architecture of InfoArchive allows the servers to be specialized for performing a specific functionality, e.g. ingestion, search etc. This allows better flexibility to handle the SLA and availability requirements (e.g. add more search server instances, to improve search responses, add dedicated servers for data ingest)

Under The Hood

InfoArchive uses XML as the format for data preservation for long-term retention, enabling platform-independence. xDB database is used to store these files (the XML data and structure), which allows xml queries to be run against the data efficiently, at any level of detail.

xDB is a proprietary, lightweight XML database. Use of XML allows encoding information for long-term, platform-independent retention.

InfoArchive allows Amultiple, different data structures to be stored in a single repository (archive), which can consist of both structured and unstructured data. The structured and unstructured data can constitute a single record (e.g. invoice content files + SAP data for invoices) and can be accessed/treated as a single object for querying/reporting.

A holding is the entity holding an archive's data (Archival Information Package or AIPs). An application can have one or more holdings. Holding can be thought of as a top level (i.e. root) folder, under which all the data resides. The data under a holding reside in flat structure.

A single instance of InfoArchive can have multiple applications, completed isolated from each other, secured by access controls.

Loading The Data

An InfoArchive holding (archive) can be either table-based or SIP-based (submission information packages). The table-based archive uses a schema to ingest structured data and linked files from a database table (not covered in this document)

For content (i.e. unstructured data), a SIP-based archive uses submission information packages (SIPs) to ingest data from files, data records, or compound records. An SIP is a ZIP file containing an index file, a descriptor file and one or more content files. The unstructured data is ingested into InfoArchive as SIPs, containing the content files, metadata XML file (containing indexes for all these content files) and a descriptor file(XML), describing the contents of the SIP.

Post upload, the SIP becomes a logical partition(AIP) in InfoArchive. The metadata is stored in xDB, and the zip file is pushed to storage location.

In case of structured data, the SIP will contain only the XML files.

Data upload can be performed using the batch jobs (scripts) or via REST calls, allowing near real-time data ingest.

Finding The Data

InfoArchive deploys a 2 step search mechanism. The logical partition(AIP) is associated with at least one search criteria. A partition would typically contain 10s of thousands of records(AIUs).

On a high level, following is the way InfoArchive search functions –

- An index is created against the search criteria, which is mapped to the specific AIP (i.e. partition key)
- 1st step search locates the AIP, which contains the record. Since AIPs will be low in numbers, this will be very quick. This search is carried out in main database, one of the databases within the xDB
- 2nd step search will perform the search within the AIP, for the specific record (Archival Information Unit or AIU). Due to smaller number of records to search from (typically multiples of 10,000), the record is located quickly

To keep the retrieval performant, a caching mechanism is supported, that allows the content to be cached in the local storage

Storage Options

InfoArchive supports a number of storage solutions, e.g. SAN, cloud, CAS. This provides the enterprise a number of options to choose from, based on the availability, performance and cost considerations. Integration with cloud storage is supported via S3 interface, allowing a standard way of integrating with different providers (both EMC ECS & Amazon S3 can be integrated using S3 interface). Cheaper storage options like Glacier allow tiered data storage, facilitating storage of low value, infrequently accessed data to be kept in a low cost storage media.

Data Security

For the data security requirements, InfoArchive supports the content and metadata encryption as well as data masking. It supports AES encryption using Bounty Castle and Java Cryptography Architecture (JCA)

As a typical use case example, personally identifiable information can be encrypted while ingesting the data. The search UI will allow authorized users to search the encrypted data. The search results will be displayed in masked format (or decrypted format for authorized groups).

Compliance

On the compliance end, InfoArchive 16.5 has built in features enabling retention/ purge management and hold capabilities. The built-in retention capabilities allow you to apply retention to records, on the application, AIP or record, using the UI/REST APIs, during ingestion or via batch jobs. Multiple retention policies can be applied to the same record. It provides the fixed, event-based and mixed mode* retention capabilities.

InfoArchive is certified against many compliance standards such as Dodd-Frank(US), MiFID2/R (EU), OFSI E-13 (Canada) etc.

*Retain the object for a fixed duration, however dispose immediately if an event implies that the object is no longer required .

Archival Scenarios and InfoArchive Fitment

Archival typically has 2 use cases - cold archiving, and live archiving.

Cold archiving includes data that is no longer part of live business processes, and its more cost effective to decommission the application containing this data. However, this data can be valuable to the business and important to preserve for compliance reasons, and hence this data set is archived, usually in one go (or in a limited, defined number of migration cycles). Access to this data is infrequent, hence performance criteria for the archival application are quite relaxed.

Live archiving refers to the use case where live business data from the leading application is archived regularly, to obtain the cost savings, and keep the leading application performant. While this data set too is read only, however the archival process is always running.

Live archiving is characterized by high ingest rate, as well as higher search/retrieval volumes compared to cold archiving, often with a custom/3rd party application accessing the InfoArchive data. Since the data is still used by the leading live application, performance and availability requirements are often quite stringent.

Below sections provide some more details on both the archival scenarios, and fitment of InfoArchive for both the cases.

Cold Archival

The traditional data archiving refers to the process of moving data from the primary application to a read-only, cost effective storage with search and retention capabilities, coupled with restricted access patterns (e.g. accessible only to administrators /limited set of users). The data is moved based on a defined set of criteria (e.g. age), which can be one time or as multiple steps process. This scenario is often associated with business application decommissioning.

Such archival applications, typically would have relaxed SLAs and availability requirements, compared to the live business applications, with data sitting in retired status, and accessed only for legal/ compliance requirements.

The underlying element of this approach is that the data has reached end-of-life from a business value perspective, and need to be retained primarily because of regulatory/ legal requirements.

For this scenario, InfoArchive is a straight fit, providing a low cost, flexible solution, with ability to store large volume of data for extended period of time, in a client agnostic format. It allows multiple archival applications to be created within a single InfoArchive instance, with options for a number of storage media (e.g. ECS, S3) to be configured. The built-in retention mechanism ensures that data is retained as per the compliance requirements, eliminating any accidental/malicious data losses. Robust data encryption capabilities ensure that sensitive data, while lying in inactive state, still has the safety measures equivalent to live business data. Role based accesses permits only the authorized users (typically a small number in this scenario) to access the data.

Live Archival

A slightly different archiving approach has increasingly found favor with large organizations (especially the financial institutions) in past decade. These companies, owing to the ever-increasing volumes of customer data, e.g. account statements, marketing e-mails, text messages, notifications etc. often look for an archival solution which provides the traditional archival cost advantages, but can also match the SLA requirements of the live business applications, along with the digital enablement for the integrations, reporting and analytics requirements.

In this scenario, typically the archival application is integrated with the leading business application, serving as its data store or sometimes directly exposed to the users via a custom search UI. Writes can happen synchronously, near-real-time (e.g. end of workflow), or asynchronously (e.g. batch jobs). Read SLAs of the applications are expected to be on par with the live business applications. Archival application availability is also critical, due to it being part of the business process.

This approach can decrease the load on a live application, increasing performance and reducing hardware costs, by offloading some/most of the application's data to InfoArchive. The data that you choose to archive, and the frequency of archiving, is based on your business rules. For example, on a weekly basis, a bank can archive all bank statements that are a year old or more

For the live archiving scenario, InfoArchive's fitment needs to be thoroughly assessed, based on the availability and SLA requirements of the application. While InfoArchive is quite performant when it comes to ingest, search or retrieval, its high availability capabilities have some limitations (the underlying single instance of xDB presents a single point of failure).

Design Considerations
Search Architecture

The InfoArchive search architecture, as described earlier, relies on a 2 step search mechanism, i.e. 1st step identifies the AIP containing the record, while the 2nd step searches for a given record within the AIP.

For the search design, the AIP size and mode should be chosen carefully, since if the AIP size is too small (e.g. 10 records per AIP), then for a large repository, the AIP count itself will be very high, and it'll impact the 1st step performance (as per OpenText, post 200,000 AIPs, search performance starts to suffer). On the other hand, a very large AIP can choke the 2nd step search, thus eliminating the advantages of the 2 step search. Also, it must be evaluated where background search can serve the purpose (e.g. scheduled reporting tasks)

For batch ingest, the private AIP mode works best (large number of records, e.g. 50,000 – 100,000, put into a single AIP, gets ingested at once). Private mode means that data becomes searchable immediately once ingestion is completed.

On the other hand, if the incoming data volumes are small (typical in live archiving scenarios), then aggregated AIP mode can be a better option (AIP is created with volume and time threshold, and incoming records are pushed into it till the threshold is reached, upon which it is closed).

For medium size data volumes (typically > 1000 to < 10,000), pooled AIP mode might be a better option (multiple AIPs share an XDB library, multiple AIPs can be open simultaneously)

From an availability perspective, since xDB is a single point of failure, search architecture should be carefully vetted against the SLA requirements. For live environments, a custom search/caching layer may need to be built on top of InfoArchive, to serve the required availability requirements. Alternatively, an active DR architecture can also be considered (with xDB replication enabled, and incremental content replication in place)

Retention Design

InfoArchive supports fixed, event based as well as Mixed Mode retention. However, some factors need to be carefully considered while choosing the retention type for the given types/objects.

Two factors need to be evaluated carefully for an effective & easy to use retention architecture, which are -

- Functionality required
- Capacity to be handled

Functionality wise, InfoArchive does have a specific behavior when used with a certain hardware combination, so if there is a need for mixed mode retention, the requirements should be thoroughly evaluated against the InfoArchive capabilities and the product stack in use.

For example, when InfoArchive is configured with ECS cloud storage, the ECS storage will apply its own hardware retention policy, on top of InfoArchive retention, based on the duration supplied by InfoArchive. ECS treats this duration as fixed, and it doesn't allow the item to be purged before this fixed date, neither does it allow the policy duration to be shortened (though it'll allow the policy duration extension). This is not the case with S3 or SAN.

Scenarios like this might have cost and possibly compliance implications in certain cases, and hence requirements should be clearly articulated early in project, and design should take into account this limitation.

Capacity wise, if the number of records is massive (e.g. billions), then the retention design should be carefully thought about, otherwise the managed item database (retention database) may become huge in size, having design and cost implications.

For example, in event based retention, a retention policy is created for each object. If we assume that objects for 1 record require 8 kb of storage space, then if event based retention is set up on 1 billion records, the resultant policy objects will require almost 7.5 TB of space. This, coupled with the fact that the managed item database needs to reside on the block storage, on a single xDB instance, may require a very large VM or physical server for xDB. This can create challenges for the infrastructure architecture & maintenance. So for very large data sets, Batch level retention policies will be better suited.

Availability

For the live archiving use cases where availability SLAs are stringent, the application design should take into consideration the InfoArchive reliance on a single xDB instance (the Web & IAS layer support high availability). There are design possibilities to mitigate this risk to a certain extent, e.g. spread the databases in xDB on different machines, so to protect against complete availability loss (e.g. the searches will still be functional if main database is up, while others are down). However even in this scenario, the availability risk will be there (e.g. the main database itself may go down, making InfoArchive metadata unavailable)

While the web UI and InfoArchive server support high availability, xDB as a single point of failure limits the overall application availability and failover options.

A possible option to handle this, can be to create an XDB replica and automate the switch from primary instance to replica upon failure detection. xDB doesn't provide any tools/mechanisms to detect primary instance failure and automatic failover, hence a custom script/tool needs to be used for this. This can provide a limited level of failover capability, however failover will not be seamless in this case as InfoArchive server needs to be rebooted for failover to take effect (i.e. to point to new instance), hence the switchover will be visible to the users.

AIP Storage and Retrieval

Like many other applications, InfoArchive's search/retrieval performance too, is greatly dependent on the solution design. As described earlier, search performance requires thoughts around data partitioning strategy.

Similarly, data retrieval performance can vary a lot, based on the AIP creation strategy (i.e. data packaging). For example, retrieving a record from a 15 GB AFP (a data format) record (i.e. the entire file in an AIP) will take considerable time, due to the need to load the AFP file on disk, then parse it, and retrieve the document.

On the other hand, if the above AFP is split per record while ingesting, the record retrieval will be very quick, however the downside will be that the storage required for the document will be much greater than the original document size (3 – 4 times).

Hence the AIP creation strategy should strive to find a balance between the 2 extremes. For example, AIPs with n*100 records in one file (where n <=50 typically offer better performance, while providing the space savings)

Encryption Design

Encryption is invaluable as a mechanism for protecting data, particularly personally identifiable information, from disclosure to unauthorized channels. For encryption to be ensure security and be effective across large enterprises, the encryption keys must be managed with the same level of attention, given to the confidential data that they protect. Enough protection should be introduced in the solution to ensure that the keys are not easily guessed, disclosed or lost, and so that the data they encrypt can be recovered by authorized channels.

InfoArchive supports a number of encryption providers (e.g. bouncy castle, JCA), offering multiple encryption options. However, it provides limited options when it comes to integrate with more sophisticated centralized encryption and key management solutions. For example, it doesn't support integration with Vormetric or Voltage Security. Gemalto-SafeNet, which offered some of these capabilities in earlier version, is not supported in 16.x versions.

Hence use of InfoArchive should take into consideration the enterprise key management solution in place if any, and the design challenges it may introduce for InfoArchive architecture

Transformation

In the Archival space, with growing number of application areas and content resources involved, the diversity challenges related to source data have also increased. This presents the heterogeneity challenges, i.e. how to present the AFP, Pdf, Word or other formats content, in a unified, read-only format to the end-user.

The source content is often varied in nature, e.g. customer statements (AFP), communications(e-mails), invoices (jpeg/tiff), manuals(pdf) or other content in different formats. To make it suitable for a support executive, to resolve a consumer query quickly, or for an analyst to derive meaningful insights from the content, the format homogeneity is often a pre-requisite.

For InfoArchive, while this is not a core product capability (except some pdf rendering features), and needs to be handled using additional solution elements, this still can have a large impact on the content storage (and thus costs), as well as retrieval performance (important in live archival scenarios).

Transformation can typically be performed in 2 different ways

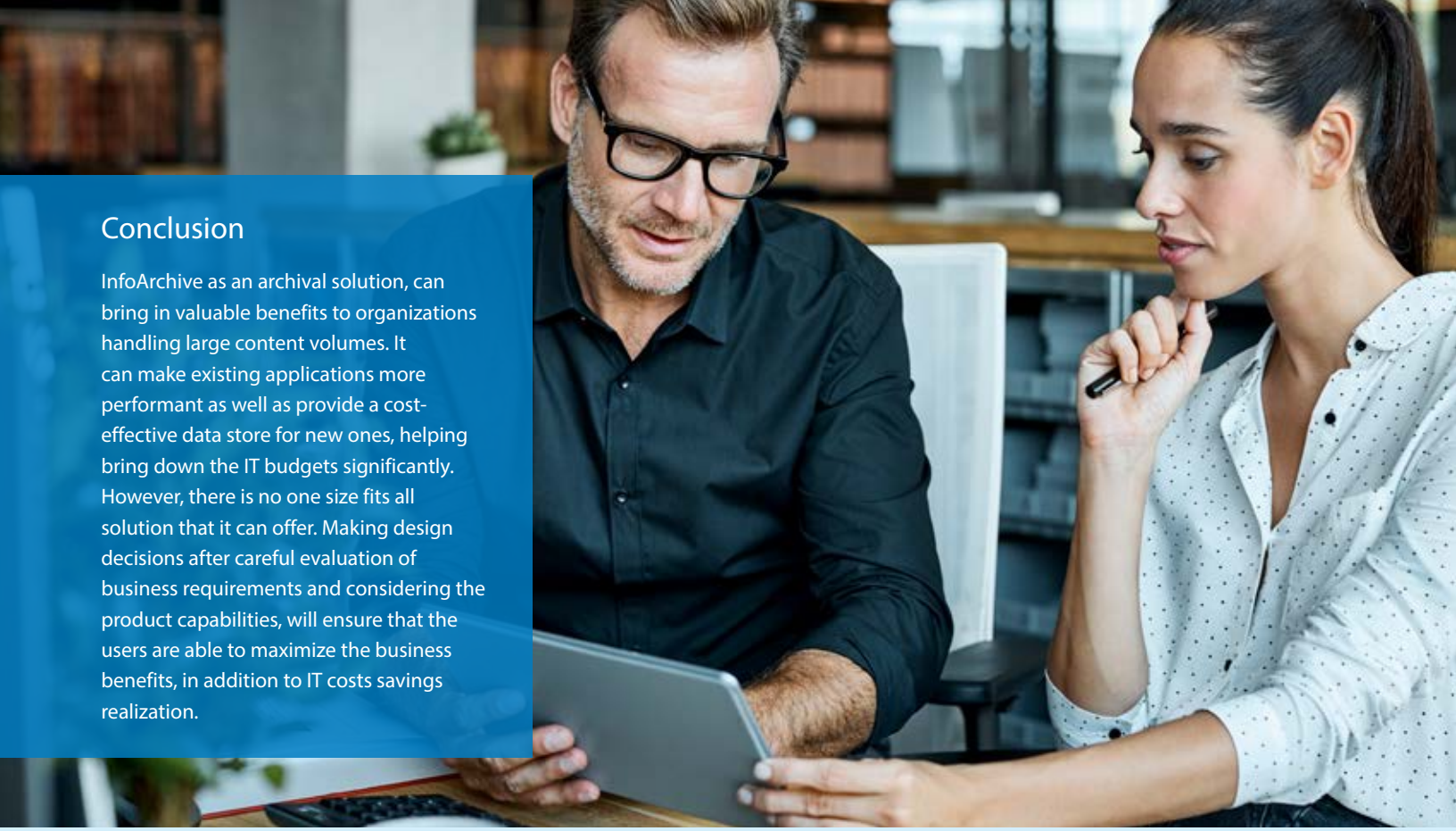
1. Ingest transformation - as the name implies, it deals with content transformation at the time of ingest
2. On-the-fly transformation - store the content in original format, transform it only at the time of retrieval

The decision in favor of either of the 2 approaches (or a combination of both) should be carefully vetted against the storage costs, retrieval performance, as well as regulatory requirements (i.e. keep the 'original' content copy).

For example, converting a 10 GB AFP to pdf while ingesting it, will reduce the size a bit, however will introduce performance challenges in terms of individual record retrieval. The same AFP, if split at each record level, will make the retrievals very efficient, however the size will increase multifold (3 to 4 times compared to original), due to duplication of resources in each pdf.

In this example, a middle approach might work better, i.e. a pdf with n number of records (where n could range from 100s to a couple of 1000s records) typically offer the performance benefits, while still not bloating the storage.

Some add-ons (e.g. ProArchiver) offer certain storage and performance advantages by offering a custom format for storage, however this needs to take into consideration the vendor lock-in, and subsequent dependency on the proprietary format.



Conclusion

InfoArchive as an archival solution, can bring in valuable benefits to organizations handling large content volumes. It can make existing applications more performant as well as provide a cost-effective data store for new ones, helping bring down the IT budgets significantly. However, there is no one size fits all solution that it can offer. Making design decisions after careful evaluation of business requirements and considering the product capabilities, will ensure that the users are able to maximize the business benefits, in addition to IT costs savings realization.

About the Author

Ravi is one of the ecm architects with Infosys, working primarily in OpenText technologies. When not solving the design challenges, he can be often found immersed in automobile materials.

For more information, contact askus@infosys.com

Infosys®
Navigate your next